INTEGRATED CIRCUITS

DATA HANDBOOK

80C51-Based 8-Bit Microcontrollers

BOOK IC20 1992

Signetics

Philips Semiconductors

PHILIPS

80C51-Based
8-Bit Microcontrollers

Signetics

Philips Semiconductors

PHILIPS

Signetics reserves the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Signetics assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified. Applications that are described herein for any of these products are for illustrative purposes only. Signetics makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. There are undocumented characteristics of some products specified in this databook. Signetics will not be responsible for any operating differences between the ROM-based and EPROM-based device of the same microcontroller.

LIFE SUPPORT APPLICATIONS

Signetics Products are not designed for use in life support appliances, devices, or systems where malfunction of a Signetics Product can reasonably be expected to result in a personal injury. Signetics customers using or selling Signetics Products for use in such applications do so at their own risk and agree to fully indemnify Signetics for any damages resulting from such improper use or sale.

Signetics registers eligible circuits under
the Semiconductor Chip Protection Act.

© 1992 Signetics Company.

All rights reserved.

Preface

80C51-Based
8-Bit Microcontrollers

Microcontrollers from Philips Semiconductors

Philips Semiconductors supplies a wide range of microcontrollers based on mainstream architectures. By offering a large variety of product derivatives, Philips Semiconductors can meet a broad range of specific or unique application requirements. All of our microcontrollers are based on mainstream architectures to allow the user to take advantage of existing software and a vast array of third-party support.

Philips Semiconductors 8-bit microcontrollers are based on the popular 80C51 and 80C49 architectures. We offer most of the industry standard products in these architectures as well as a large selection of powerful derivative products. These derivatives offer a wide assortment of features, including: additional memory, A/D, PWM, additional timers, and many more. Many of the derivative microcontrollers have an I^2C serial interface that allows them to be connected easily to over 70 other parts, increasing their capabilities even further. The I^2C serial bus is covered in Section 2 of this book. This data book covers the 80C51 standard products and derivatives that Philips Semiconductors manufactures. The 80C49 products are covered in a separate book.

Philips Semiconductors 16-bit microcontroller family is based on the powerful 68000 architecture. While these are called 16-bit microcontrollers, the 68000 CPU core architecture is 32-bit. This offers the user a great deal more processing power, when the need arises in a design to move from an 8-bit to a 16-bit microcontroller. Philips Semiconductors 16-bit microcontrollers are software compatible with existing 68000 code. As with our popular 8-bit microcontrollers, EPROM and OTP versions of our 16-bit products are available. The 16-bit microcontrollers are also covered in a separate data book.

Philips Semiconductors is developing a family of 32-bit microcontrollers based on the SPARC RISC architecture. This family of microcontrollers will offer the ultimate in processing power for those applications that are computation intensive in an embedded control environment.

Philips Semiconductors offers uncompromising quality, service, and support with all of its microcontroller products. For a complete family and the best in microcontroller products, look to Philips Semiconductors.

Product Status

80C51-Based
8-Bit Microcontrollers

DEFINITIONS		
Data Sheet Identification	**Product Status**	**Definition**
Objective Specification	Formative or In Design	This data sheet contains the design target or goal specifications for product development. Specifications may change in any manner without notice.
Preliminary Specification	Preproduction Product	This data sheet contains preliminary data and supplementary data will be published at a later date. Signetics reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.
Product Specification	Full Production	This data sheet contains Final Specifications. Signetics reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.

Contents

80C51-Based
8-Bit Microcontrollers

Preface iii

Product Status iv

Ordering Information ix

80C51 Microcontroller Family Features Guide x

8051 Microcontroller Cross-Reference Guide xii

80C51 Microcontroller Development System Support xiii

Signetics Microcontroller Bulletin Boards xiv

CMOS and NMOS 8-bit Microcontroller Family xv

CMOS 16-bit Microcontroller Family xxi

Philips 8-bit Microcontroller Demonstration and Evaluation Boards xxii

Section 1 – 80C51 Family

80C51 Family Overview 3
80C51 Family Architecture 8
80C51 Family Hardware Description 23
80C51 Family Programmer's Guide and Instruction Set 48
80C51 Family EPROM Products 103
8031AH/8051AH Data Sheet 107
80C31/80C51/87C51 Data Sheet 116

Section 2 – Inter-Integrated (I^2C) Circuit Bus

I^2C Bus Specification 134
I^2C Address Allocation Table 154
I^2C Bus Addresses 155
I^2C Peripheral Selection Guide 156

Section 3 – ACCESS.bus™ Technical Overview

Introduction 159
HOW ACCESS.bus Works 162
Application Device Types 166
Timing Rules 167
Microcontrollers 167
Software Architecture and Development 168
Development Support 169

Section 4 – 80C51 Family Derivatives

80CL31/80CL51 Overview 173
80CL31/80CL51 Data Sheet 177
8XC52 Overview 196
8032AH/8052AH Data Sheet 206
80C32/80C52/87C52 Data Sheet 215
8XC053/54 Overview 230
83C053/83C054/87C054 Data Sheet 234
8XCL410 Overview 250
80CL410/83CL410 Data Sheet 256
8XC451 Overview 277
80C451/83C451/87C451 Data Sheet 280
8XC524/8XC528 Overview 297
83C524/87C524 Data Sheet 302
80C528/83C528/87C528 Data Sheet 320
8XC550 Overview 340
80C550/83C550/87C550 Data Sheet 345

Contents (Continued)

Section 4 – 80C51 Family Derivatives (continued)
8XC552/562 Overview ... 363
80C552/83C552/87C552 Data Sheet ... 422
80C562/83C562 Data Sheet ... 444
87C575 Overview ... 457
80C575/83C575/87C575 Data Sheet ... 461
80C592/83C592/87C592 Data Sheet ... 480
8XC652/654 Overview ... 518
80C652/83C652/87C652 Data Sheet ... 522
83C654/87C654 Data Sheet ... 541
80CE654/83CE654 Data Sheet ... 560
8XC751 Overview ... 573
83C751/87C751 Data Sheet ... 580
8XC752 Overview ... 591
83C752/87C752 Data Sheet ... 596
8XC851 Overview ... 608
80C851/83C851 Data Sheet ... 610
83C852 Data Sheet ... 624

Section 5 – Application Notes
AN408 80C451 Operation of Port 6 ... 665
AN417 256k Centronics Printer Buffer Using the 87C451 Microcontroller ... 676
AN418 Counter/Timer 2 of the 83C552 Microcontroller ... 689
AN420 Using up to 5 External Interrupts on 80C51 Family Microcontrollers ... 696
AN422 Using the 8XC751 Microcontroller as an I²C Bus Master ... 698
AN423 Software Driven Serial Communication Routines for the 83C751 and 83C752 Microcontrollers ... 716
AN424 8051 Family Warm Boot Determinations ... 722
AN425 Interfacing the PCD8584 I²C-bus Controller to 80C51 Family Microcontrollers ... 724
AN426 Controlling Aire Core Meters with the 87C751 and SA 5775 ... 744
AN427 Timer I in Non-I²C Applications of the 83/87C751/752 Microcontrollers ... 759
AN428 Using the ADC and PWM of the 83C752/87C752 ... 765
AN429 Airflow Measurement Using the 83/87C752 and "C" ... 772
AN431 9XC1XX Philips 16/32-Bit Microcontroller Series ... 792
AN432 Parallel Port on the SBE 68070 ... 796
AN436 "Opti-Mizer" Power Management for Notebook Computers Using the 8XC752 Microcontroller ... 802
AN438 I²C Routines for 8XC528 ... 812
AN442 (BCM) 87C751 Specification for a Bus-Controlled Monitor ... 850

Section 6 – Development Support Tools
Development support tools ... 869
NOHAU EMUL51-PC – PC-Based In-Circuit Emulator ... 876
Metalink ... 886
DB-51 CEIBO Development Board ... 890
SDS 8051 Stand-Alone Debug Station for 80C51/8051-Based Systems ... 892
OM4142 (ASM51) Cross-Assembler Package for 80C51/8051-Based Systems ... 898
OM4144 (PLMTI51) PL/M-51 Compiler Package for 80C51/8051-Based Systems ... 900
OM4136 8051 C Cross-Compiler ... 903
OM4129 Symbolic Debugging Package XRAY51 for the SDS 8051 Emulator ... 905

Section 7 – Additional Microcontroller Group Data Sheets
SCN8049 SERIES (SCN8049, SCN8050, SCN8039, SCN8040) Data Summary ... 911
8X305 Data Summary ... 912
8X401 Data Summary ... 913
PCA82C200 Data Sheet ... 914
UAA1300 Data Sheet ... 954
PCF1252-X Family Data Sheet ... 969

Contents (Continued)

Section 8 – Package Outlines

24-Pin (300 mils wide) Ceramic Dual In-Line with Quartz Window (F) Package ... 981
24-Pin (300 mils wide) Plastic Dual In-Line (N) Package ... 982
28-Pin (600 mils wide) Ceramic Dual In-Line with Quartz Window (F) Package ... 983
28-pin (600 mils wide) Plastic Dual In-Line (N) Package ... 984
SOT117 28-Pin Plastic Dual In-Line (N/P) Package ... 985
28-Pin Plastic Leaded Chip Carrier (A) Package ... 986
SOT136A 28-Pin Plastic SO (Small Outline) Dual In-Line (D/T) ... 987
40-Pin (600 mils wide) Ceramic Dual In-Line with Quartz Window (F/FA) Package ... 988
40-Pin (600 mils wide) Plastic Dual In-Line (N) Package ... 989
SOT129 40-Pin Plastic Dual In-Line (P/N) Package ... 990
SOT158A 40-Pin Plastic VSO (Very Small Outline) Dual In-Line (D/T) Package ... 991
44-Pin Plastic Leaded Chip Carrier (A) Package ... 992
44-Pin Square Ceramic Leaded Chip Carrier with Quartz Window (L) Package ... 993
44-Pin Cerquad J-Bend with Quartz Window (K/KA) Package ... 994
44-Pin Plastic Quad Flat Pack (B) (MEC) Package ... 995
SOT311 44-Pin Square Plastic Quad Flat Pack (B) Package ... 996
64-Pin Plastic Dual In-Line (N) Package ... 997
68-Pin Plastic Leaded Chip Carrier (A) Package ... 998
68-Pin Square Ceramic Leaded Chip Carrier with Quartz Window (L) Package ... 999
68-Pin Cerquad J-Bend with Quartz Window (K/KA) Package ... 1000
SOT219 80-Pin Plastic Quad Flat Pack (B) Package ... 1001
SOT318 80-Pin Plastic Quad Flat Pack (B) Package ... 1002

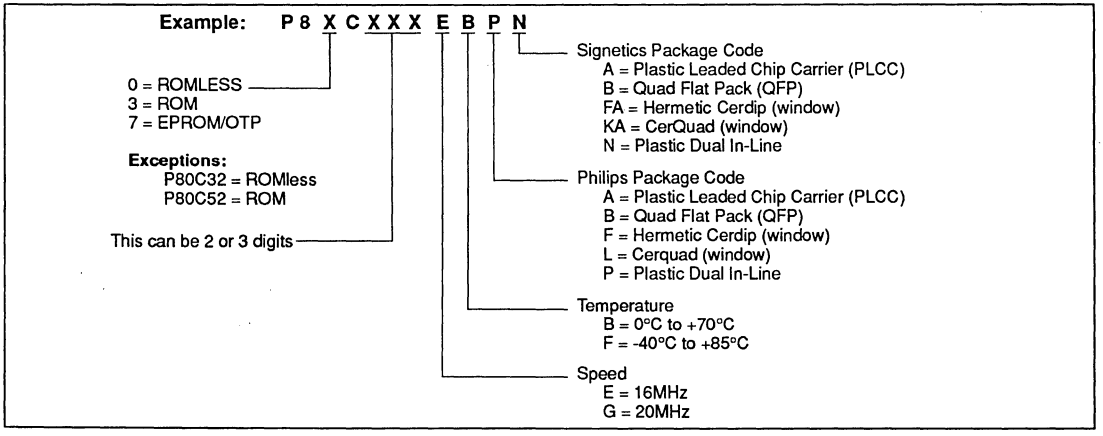
Section 9 – Sales Offices, Representatives and Distributors ... 1005

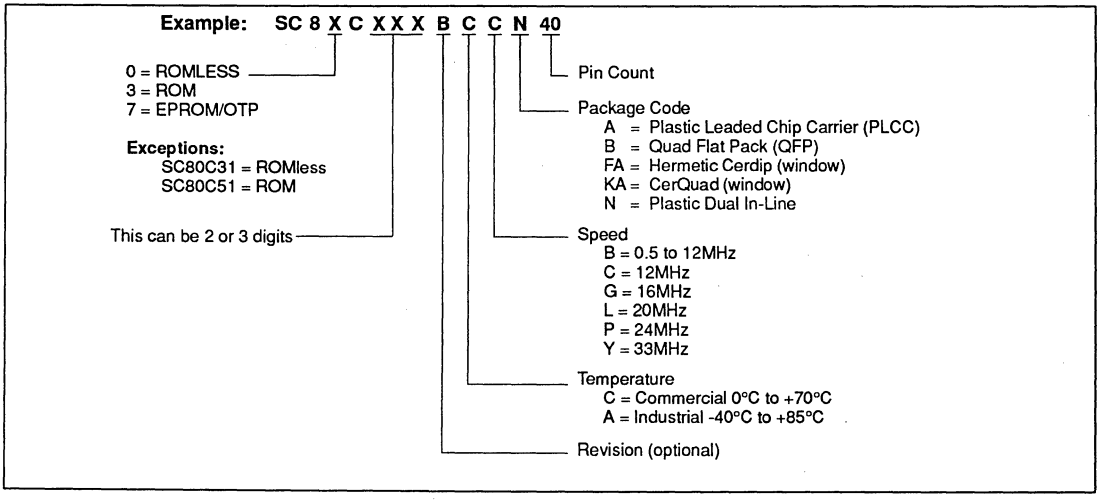
APPENDIX A: Philips Semiconductors Data Handbook System

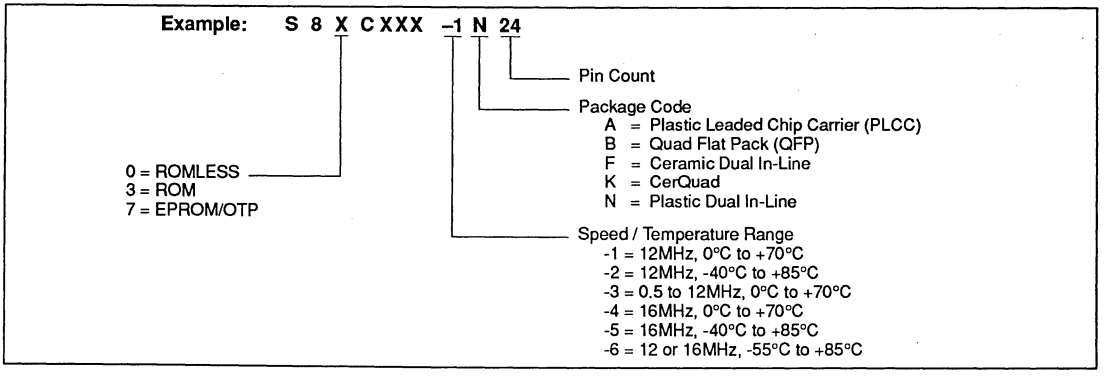
APPENDIX B: Pin Configurations

Ordering Information

MICROCONTROLLER PRODUCTS







80C51 microcontroller family features guide

PART NO.			MEMORY		COUNTER/	I/O	SERIAL	EXTERNAL	
ROMless	ROM	EPROM	ROM	RAM	TIMER	PORT	INTERFACES	INTERRUPTS	SPECIAL FEATURES
–	83C751	87C751	2k	64	1 (16-bit)	2–3/8	I²C (Bit)	2	Low-cost 24-pin skinny-DIP
–	83C752	87C752	2k	64	1 (16-bit)	2–5/8	I²C (Bit)	2	5-Channel 8-bit A/D, PWM output
8031AH	8051AH	–	4k	128	2	4	UART	2	NMOS
80C31B	80C51B	87C51	4k	128	2	4	UART	2	CMOS
80CL31	80CL51	–	4k	128	2	4	UART	10	Low voltage (1.8V–6V), low power
80CL410	83CL410	–	4k	128	2	4	I²C	10	Low voltage (1.8V–6V), low power
80C451	83C451	87C451	4k	128	2	7	UART	2	Extended I/O, Processor bus interface
80C550	83C550	87C550	4k	128	2 + Watchdog	4	UART	2	8-Channel 8-bit A/D
80C851	83C851	–	4k	128	2	4	UART	2	256 Bytes EEPROM, 8051 pin-for-pin compatible
–	83C852	–	6k	256	2	2/8	–	–	Smartcard microcontroller 2k EEPROM (data, code) Calculation unit for cryptographic calculations.
8032AH	8052AH	–	8k	256	3	4	UART	2	NMOS
80C32	80C52	87C52	8k	256	3	4	UART	2	CMOS
80C575	83C575	87C575	8k	256	3 + PCA + Watchdog	4	Enhanced UART	2	4 analog comparators, Low Active Reset
80C552	83C552	87C552	8k	256	3 + Watchdog	6	UART, I²C	6	8-Channel 10-bit A/D, 2 PWM outputs
80C562	83C562	–	8k	256	3 + Watchdog	6	UART	6	8-Channel 8-bit A/D, 2 PWM outputs
80C652	83C652	87C652	8k	256	2	4	UART, I²C	2	8051 pin-for-pin compatible with twice the memory
–	83C053	–	8k	192	2 (16-bit)	3-4/8	–	3	On-screen display, 9 PWM outputs, 3 software A/D inputs
–	83C054	87C054	16k	192	2 (16-bit)	3-4/8	–	3	On-screen display, 9 PWM outputs, 3 software A/D inputs
–	83C654	87C654	16k	256	2	4	UART, I²C	2	8051 pin-for-pin compatible with four times the ROM and twice the RAM
–	83CE654	–	16k	256	2	4	UART, I²C	2	83C654 with Electromagnetic Compatibility improvements
80C592	83C592	87C592	16k	512	3+ Watchdog	6	CAN + UART	6	8-Channel 10-bit A/D, 2 PWM outputs, CAN bus interface
80C528	83C528	87C528	32k	512	3 + Watchdog	4	UART, I²C (Bit)	2	The Ultimate Programming Machine

80C51 microcontroller family features guide

PART NO.	SPEED					TEMPERATURE °C			PACKAGE					
	0.5–12	12	16	20/24	33	0 to 70	–40 to +85	–55 to +125	PDIP	CDIP	PLCC	CLCC	QFP	VSO
83/87C751	X	X	X			X	X	X (–40 to +125)	24	24	28			
83/87C752	X	X	X			X	X	X	28	28	28			
8031/51		X	X (15MHz)			X	X		40		44			
80C31/51/87C51	X	X	X	X	X	X	X	X	40	40	44	44	44	
80CL31/51			X (32kHz–16MHz)				X		40					40
80/83CL410			X (32kHz–16MHz)				X		40					40
80/83/87C451	X	X	X			X	X	X	64	64	68	68		
80/83/87C550		X	X			X	X	X	40	40	44	44		
80/83C851		X				X	X		40		44		44	
83C852		X				X								
8032/52		X	X (15MHz)			X	X		40		44			
80C32/52/87C52			X	X		X	X	X	40	40	44	44	44	
80/83/87C575			X	X		X	X		40	40	44	44	44	
80/83/87C552			X	X		X	X	X			68	68	80	
80/83C562			X			X		X (–40 to+125)			68			
80/83/87C652			X	X		X	X	X (–40 to +125)	40	40	44	44	44	
83C053		X				X			42 SDIP					
83/87C054		X				X			42 SDIP					
83/87C654			X	X		X	X	X (–40 to +125)	40	40	44	44	44	
83CE654			X			X	X						44	
80/83/87C592			X					X (–40 to +125)			68	68		
80/83/87C528			X	X		X	X	X (–40 to +125)	40	40	44	44	44	

NOTE:
All combinations of part type, speed, temperature, and package may not be available.

8051 microcontroller cross-reference guide

	INTEL	AMD	SIEMENS	OKI	MATRA/HARRIS	PHILIPS SEMICONDUCTORS/ SIGNETICS
NMOS	8039AL 8049AH 8040AHL 8050AH					MAB8039H/SCN8039H MAB8049H/SCN8049H MAB8040H/SCN8040H MAB8050H/SCN8050H
	8031AH 8051AH 8032AH 8052AH	8031AH 8051AH	SAB 8031A SAB 8051A SAB 8032A SAB 8052A			MAB8031AH/SCN8031H MAB8051AH/SCN8051H MAB8032AH/SCN8032H MAB8052AH/SCN8052H
CMOS	80C31BH 80C31BH-1 80C31BH-2	80C31BH	SAB 80C31	MSM80C31 MSM80C31	80C31 80C31-1 80C31	PCB80C31BH-2/SC80C31BCC PCB80C31BH-3/SC80C31BCG /SC80C31BCB
	80C51BH 80C51BH-1 80C51BH-2	8051BH	SAB 80C51	MSM80C51 MSM80C51	80C51 80C51-1 80C51	PCB80C51BH-2/SC80C51BCC PCB80C51BH-3/SC80C51BCG /SC80C51BCB
	87C51 87C51-1 87C51-2	87C51				SC87C51CC SC87C51CG SC87C51CB
	80C32 80C32-1 80C52 80C52-1	80C52T2				P80C32EB P80C32GB P80C52EB P80C52GB

NOTES:
1. Siemens 8032A–16 = 16MHz 8032.
2. AMD 80C52T2 = 80C52 without T2.
3. 80XXAHL = 80XX with low power standby pin; H = HMOS.

80C51 microcontroller development system support

DEVELOPMENT SYSTEM CONTACTS

COMPANY	ADDRESS	TELEPHONE
Ashling Microsystems Limited	Plassey Technological Park Limerick, Ireland	(353) 63-334466
BSO Tasking	128 Technology Center P.O. Box 9164 Waltham, MA 02254-9164	(617) 894-7800
Ceibo Ltd.	105 Gleason Rd. Lexington, MA 02173 Merkazim Building, Industrial Zone P.O. Box 2106 Herzelia 46120, ISRAEL	(617) 863-9927 972-52-555387
Nohau Corp.	51 E. Campbell Ave. Campbell, CA 95008	(408) 866–1820
MetaLink Corp.	325 E. Elliot Road, Suite 23 Chandler, AZ 85225	(602) 926–0797
Philips Semiconductors	Corporate Centre Building BAE-2 P.O. Box 218 5600 MD Eindhoven The Netherlands	31-40-724223
SIGNUM Systems	171 E. Thousand Oaks Blvd., #202 Thousand Oaks, CA 91360	(805) 371-4608

EPROM PROGRAMMING SUPPORT CONTACTS

Advin Systems 1050-L East Duane Ave. Sunnyvale, CA 94086 (408) 736-2503	Logical Devices, Inc. 1201 Northwest 65th Place Ft. Lauderdale, FL 33309 (305) 974-0967	North Valley Products P.O. Box 32899 San Jose, CA 95152 (408) 929-5345
BP Microsystems 10681 Haddington #190 Houston, TX 77043 (800) 225-2102, (713) 461-9430	Logical Systems P. O. Box 6184 Syracuse, NY 13217-6184 (315) 478-0722	Strebor Data Communications 1008 N. Nob Hill American Fork, UT 84003 (801) 756-3605
Data I/O Corp. 10525 Willows Road N.E. P.O. Box 97046 Redmond, WA 98073-9746 (206) 867-6899	Needham's Electronics 4535 Orange Grove Ave. Sacramento, CA 95841 (916) 924-8037	

SOFTWARE SUPPORT CONTACTS

COMPANY	ADDRESS	TELEPHONE
Franklin Software, Inc.	888 Saratoga Ave. #2 San Jose, CA 95129	(408) 296–8051
Archimedes Software, Inc.	2159 Union St. San Francisco, CA 94123	(415) 567–4010
BSO/Tasking	Tasking Software BV P.O. Box 899 3800 AW Amersfoort The Netherlands BSO Tasking 128 Technology Center P.O. Box 9164 Waltham, MA 02254-9164	31-33-55-85-84 (Telephone) 31-33-55-00-33 (Fax) (617) 894-7800 (Telephone) (617) 894-0551 (Fax) (710) 324-0760 (Telex) (800) 458-8276 (Toll Free)

NOTE:
For more information on Development Support, see Section 6 of this book.

Signetics microcontroller bulletin boards

To better serve our customers, Signetics maintains a microcontroller bulletin board. This computer bulletin board system features a microcontroller newsletter, application and demonstration programs for download, and the ability to send messages to microcontroller application engineers. The system can be accessed with a modem at 2400, 1200, or 300 baud.

The telephone numbers are:

(800) 451-6644 (in the U.S.)
or
(408) 991-2406

Please call us anytime!

We also have a ROM code bulletin board through which you can submit ROM codes. This is a closed bulletin board for security reasons. To get an ID, contact your local sales office. The system can be accessed with a 2400, 1200, or 300 baud modem, and is available 24 hours a day.

The telephone number is:

(408) 991-3459

CMOS and NMOS 8-bit microcontroller family

80C51 FAMILY CMOS

TYPE	ROM/ EPROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	PROBE SDS	METALINK EMULATOR	REMARKS
80C31 80C51 87C51	0 4k ROM 4k EPROM	128 128 128	33 33 33	DIL40, LCC44 QFP44	UART, 2 timers		OM1092 + OM1097 (16MHz)	SUI-8051SD	OM1092: Universal probe OM1095: Upgrade unit
80C32 80C52 87C52	0 8k ROM 8k EPROM	256 256 256	20 20 20	DIL40, LCC44 QFP44	UART, 3 timers		OM4111 + OM4110	SUI-8051SD	
80C451 83C451 87C451	0 4k ROM 4k EPROM	128 128 128	16 16 16	DIP64/LCC68	UART, 2 timers Extended I/O		OM4123	SMI-83C451SD SMI-80C451SD	OM4124: PLCC to DIL OM4125: DIL to PLCC
83C528 87C528	32k ROM 32k EPROM	512 512	16 16, 20	DIL40/LCC44 (QFP44)	UART, 3 timers Watchdog timer Bit I^2C		OM4111 + OM4110		OM4110: gen. probe OM4111: probe head OM4120-S for max. speed
83C550 87C550	4k ROM 4k EPROM	128 128	16 16	LCC44 DIL44	UART, 2 timers 8 8-bit ADC inputs, watchdog timer		OM5055		
80C552 83C552 87C552	0 8k ROM 8k EPROM	256 256 256	16, 24 16, 24 16	LCC68/QFP80	UART, 2 timers Timer with compare and capture, 2 PWM outputs, 8 10-bit ADC inputs, Byte I^2C		OM1092 + OM1095	SMI-80C552SD	OM1092: Universal probe OM1095: Upgrade unit
80C562 83C562	0 8k ROM	256 256	16 16	LCC68/QFP80	UART, 2 timers Timer with compare and capture, 2 PWM outputs, 8 8-bit ADC inputs	Use 87C552 for development	OM1092 + OM1095	SMI-80C552SD	OM1092: Universal probe OM1095: Upgrade unit
80C575 83C575 87C575	0 8k 8k EPROM	256	16	DIL40, LCC44 QFP44	Enhanced UART, PCA, 4 analog comparators	80/83C575 Production: August '92			
80C592 83C592 87C592	0 16k ROM 16k EPROM	512 512 512	16 16 16	LCC68/QFP80	8XC552 + CAN interface No I^2C	Samples: Q4 '91 Production: Q1 '92	OM4110 + OM4112		OM4110: gen. probe OM4112: probe head OM4120S: full speed
80C652 83C652 87C652	0 8k ROM 8k EPROM	256 256 256	16, 24 16, 24 16, 20	DIL40/LCC44 QFP44	UART, 2 timers Byte I^2C		OM1092 + OM1096	SMI-80C652SD	

CMOS and NMOS 8-bit microcontroller family

80C51 FAMILY CMOS (Continued)

TYPE	ROM/ EPROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	PROBE SDS	METALINK EMULATOR	REMARKS
83C654 87C654	16k ROM 16k EPROM	256 256	16,24 16,20	DIL40/LCC44 QFP44	UART, 2 timers Byte I^2C		OM1092 + OM1096		OM1092: Universal probe OM1095: Upgrade unit
83CE654	16k ROM	256	16	QFP44	UART, 2 timers Byte I^2C	83C654 with Electromagnetic Compatibility improvements	OM1092 + OM1096		OM1092: Universal probe OM1095: Upgrade unit
83C751 87C751	2k ROM 2k EPROM	64 64	16 16	DIP24 skinny LCC28 DIP24 skinny	1 timer Bit I^2C		OM1094P	SMI-80C751SD	
83C752 83C752	2k ROM 2k EPROM	64 64	16 16	DIP28, LCC28 DIP 28, LCC28	1 timer, PWM output, 5 8-bit ADC inputs, Bit I^2C		OM5072	SMI-83C752SD	
80C851 83C851	0 4k ROM	128 128	16 16	DIL40/LCC44 QFP44	UART, 2 timers 256 byte	QFP44 In devel.	OM1092		
83C852	6k ROM	256	16		2k byte EEPROM smart card hardware CU		OM4119		
83C053	8k ROM	192	12	DIP42 Shrunk	2 timers, 14-bit PWM, 8-6 bit PWM 128 char. OSD 3 4-bit A/D inp.		OM5054		
83C054 87C054	16k ROM 16k EPROM	192 192	12 12	DIP42 Shrunk DIP42 Shrunk	As 8XC053		OM5054		

* The following microcontollers have no external memory access: 8XC751, 8XC752, 8XC053, 87C054, 83C852.

80CLXXX FAMILY CMOS

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	PROBE SDS	REMARKS
85CL001	0	256	16	PLCC84	UART, 2 timers Byte I^2C			
85CL000	0	256	16	DIL40	UART, 2 timers Byte I^2C		OM1079	Piggyback
83CL410	4k	128	16	DIL40 VSO40	2 timers Byte I^2C		OM1079	
83CL710	16k	256	16	DIL40	UART, 2 timers Byte I^2C	In devel.	OM1079	
80CL51	4k	128	16	DIL40 VSO40	UART, 2 timers		OM1079	
83CL580	6k	256	16	VSO56 QFP64	AD converter, PWM, UART, WD	In devel.		

CMOS and NMOS 8-bit microcontroller family

8051 FAMILY NMOS

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	PROBE SDS	METALINK EMULATOR	REMARKS
8051 8031	4k 0	128 128	15 15	DIL40/PLCC44 DIL40/PLCC44	UART, 2 timers		OM1091 + OM1097	SUI-8051SD	
8052 8032	8k 0	256 256	15 15	DIL40/PLCC44 DIL40/PLCC44	UART, 3 timers UART, 3 timers		OM4111 + OM4110	SUI-8051SD	

8048 FAMILY NMOS

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE
8048 8035	1k 0	64 64	11 11	DIL40/PLCC44 DIL40/PLCC44
8049 8039	2k 0	128 128	11 11	DIL40/PLCC44 DIL40/PLCC44
8050 8040	4k 0	256 256	11 11	DIL40/PLCC44 DIL40/PLCC44

8048 FAMILY CMOS

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE
80C49 80C39	2k 0	128 128	15 15	DIL40/PLCC44 DIL40/PLCC44

CMOS and NMOS 8-bit microcontroller family

8400 FAMILY CMOS

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	PROBE SDS	REMARKS
84C21 84C41 84C81 84C41C	2k 4k 8k 4k	64 128 256 128	10 10 10 12	DIL28/SO28 DIL28/SO28 DIL28/SO28 DIL28/SO28	20 I/O lines 8-bit timer Byte I^2C		OM1083	
84C12 84C22 84C42 84C12A	1k 2k 4k 1k	64 64 64 64	10 10 10 16	DIL20/SO20 DIL20/SO20 DIL20/SO20 DIL20/SO20	13 I/O lines 8-bit timer		OM1083	
84C00B 84C00T	0 0	256 256	10 10	28 pins VSO-56	20 I/O lines 8-bit timer Byte I^2C	Piggyback ROMless	OM1080	
84C121 84C121B	1k 0	64 64	10 10	DIL20/SO20	13 I/O lines 2 8-bit timers 8 bytes EEPROM	 Piggyback	OM1073	
84C122	1k	32	10	SO20/SO24	Controller for remote control			
84C230	2l	64	10	DIL40/VSO40	12 I/O lines 8-bit timer 16*4 LCD drive		OM1072	
84C430 84C430B	4k 0	128 128	10 10	QFP64	24 I/O lines 8-bit timer Byte I^2C 24*4 LCD drive	 Piggyback for C230 and C430	OM1072	
84C633 84C633B	6k 0	256 256	16 16	VSO56	28 I/O lines 8-bit timer 16-bit up/down counter 16-bit timer with compare and capture 16*4 LCD drive		OM1086	
84C440 84C441 84C640 84C641 84C643 84C644 84C840 84C841 84C843 84C844	4k 4k 6k 6k 6k 6k 8k 8k 8k 8k	128 128 128 128 128 128 192 192 192 192	10 10 10 10 10 10 10 10 10 10	DIP42 shrunk	RC: 29 I/O lines LC: 28 I/O lines 8-bit timer 1 14-bit PWM 5 6-bit PWM 3-bit ADC OSD 2L-16	I^2C, RC I^2C, LC I^2C, RC I^2C, LC RC LC I^2C, RC I^2C, LC RC LC	OM1074	For emulation of LC versions, use OM1074 + adapter_3
84C646 84C846	6k 8k	192 192	10 10	DIP42 shrunk	tbf	I^2C, RC I^2C, RC	tbf	
84C647 84C847	6k 8k	192 192	10 10	DIP42 shrunk	tbf	I^2C, LC I^2C, LC	tbf	

CMOS and NMOS 8-bit microcontroller family

8400 FAMILY CMOS (Continued)

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	PROBE SDS	REMARKS
84C85	8k	256	10	DIL40/VSO40	32 I/O lines 8-bit timer Byte I^2C		CM1070	
84C85B	0	256	10			Piggyback for C85		
84C853	8k	256	16	DIL40/VSO40	33 I/O lines 8-bit timer 16-bit up/down counter 16-bit timer with compare and capture		OM1081	
84C853B	0	256	16			Piggyback for C853		
84C270	2k	128	10	DIL40/VSO40	8 I/O lines 16*8 capture keyboard matrix 8-bit timer		OM1077	
84C470	4k	128	10	DIL40/VSO40				
84C270B	0	128	10			Piggyback for C270		
84C470B	0	128	10		470 also handles mech. keys	Piggyback for C470		
84C271	2k	128	10	DIL40	8 I/O lines 16*8 mech. keyboard matrix 8-bit timer		OM1078	

8400 FAMILY NMOS

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	EMULATOR TOOLS	REMARKS
8411	1k	64	6	DIL28/SO28	20 I/O lines 8-bit timer Byte I^2C		OM1084	
8421	2k	64	6	DIL28/SO28				
8441	4k	128	6	DIL28/SO28				
8461	6k	128	6	DIL28/SO28				
8422	2k	64	6	DIL20	13 I/O lines 8-bit timer Bit I^2C		PM8327/20 + PM8447	On PMDS
8442	4k	128	6	DIL20				
8401B	0	128	6	28-pin PLCC68		Piggyback for 84X1 Bond out		
8401WP	0	128	6					

CMOS and NMOS 8-bit microcontroller family

3300 FAMILY CMOS

TYPE	ROM	RAM	SPEED (MHz)	PACKAGE	FUNCTIONS	REMARKS	PROBE SDS	REMARKS
3315	1.5k	160	10	DIL28/SO28	20 I/O lines 8-bit timer V_{DD} > 1.8V		OM1083	
3343	3k	224	10	DIL28/SO28	20 I/O lines 8-bit timer V_{DD} > 1.8V Byte I^2C		OM1083	
3344	2k	224	3.58	DIL28/SO28	20 I/O lines 8-bit timer DTMF generator		OM1071	
3346	4k	128	10	DIL28/SO28	20 I/O lines 8-bit timer Byte I^2C 256 bytes EEPROM V_{DD} < 1.8V		OM1076	
3347	1.5k	64	3.58	DIL20/SO20	12 I/O lines 8-bit timer DTMF generator		OM1071 + Adapter_2	
3348	8k	256	10	DIL28/SO28	20 I/O lines 8-bit timer Byte I^2C V_{DD} < 1.8V		OM1083	
3349	4k	224	3.58	DIL28/SO28	20 I/O lines 8-bit timer DTMF generator		OM1071	
3350A	8k	128	3.58	VSO64	30 I/O lines 8-bit timer DTMF generator 256 bytes EEPROM	To be developed		
3351A	2k	64	3.58	DIL28/SO28	20 I/O lines 8-bit timer DTMF generator 128 bytes EEPROM		OM5000	
3352A	6k	128	3.58	DIL28/SO28	20 I/O lines 8-bit timer DTMF generator 128 byte EEPROM		OM5000	
3301B						Piggyback for 3315, 3343, 3348	OM1083	
3344B						Piggyback for 3344, 3347, 3349	OM1071	
3346B						Piggyback for 3346	OM1076	

CMOS 16-bit microcontroller family

16-BIT CONTROLLERS

TYPE	(EP)ROM	RAM	SPEED (MHz)	FUNCTIONS	REMARKS	DEVELOPMENT TOOLS
68070	–	–	17.5	2 DMA channels, MMU, UART, 16-bit timer, I²C, 68000 bus interface, 16Mb address range		OM4160 Microcore OM4161 (SBE68070) TRACE32-ICE68070 (Lauterbach) OM4222 90C Development system (planned)
93C101	34k	512	15	Derivative with low power modes	Low power micro-controller	
90C100 93C100 97C100	– 34k 32k (EPROM)	512 512 512	15 15 15	UART, I²C, 3 16-bit timers, 80C51 interface, 68000 interface, 40 I/O lines, 2Mb address range		OM4160/3 Microcore 3 OM4201WP (SBE90C110) OM4220 90C Development system TRACE32 – ICE93C110 (Lauterbach)

16-BIT MICROCONTROLLER FAMILY[1]

PART NO.	ROM	RAM	EEPROM	16-BIT I/O PORTS	SERIAL I/O	DMA CHANNELS	COUNTER/ TIMER	EXTERNAL INTERRUPTS	SPEED MHz	PACKAGES	SPECIAL FEATURES
68070	–	–	–	–	UART, I²C	2	1[2]	6	10, 12, 15, 17.5	PLCC84 QFP120	Memory management unit 68000 bus interface
90C100	–	512	–	2 + 1/2	UART, I²C	–	1[2]	8	15	PLCC84 QFP80	80C51 bus interface 68000 bus interface
93C100	34k	512	–	2 + 1/2	UART, I²C	–	1[2]	8	15	PLCC84 QFP80	80C51 bus interface 68000 bus interface
97C100	32k EPROM	512	–	2 + 1/2	UART, I²C	–	1[2]	8	15	PLCC84 CLCC84 QFP80	80C51 bus interface 68000 bus interface

NOTES:
1. 68000 software compatible.
2. 16-bit timer with two match/count/capture registers.

Philips 8-bit microcontroller demonstration and evaluation boards

I^2C demonstration board based on 84CXX derivatives (OM4151)

I^2C LCD driver demonstration board with 84CXX microcontroller

I^2C bus analyzer (with 84CXX)

I^2C demonstration board based on 80C51 derivatives (S87C00K)

8051 family demonstration board (OM4238, P8051DB)

8XC552 evaluation board (OM4128)

CAN controller evaluation board (OM4130, PCAN-EVAL)

8XC592 evaluation board (OM4239)

68070 demonstration and evaluation board MicroCore I (OM4160, SM68070)

90C100 family demonstration and evaluation board MicroCore III (OM4160/3, SM90C100)

Section 1
80C51 Family Overview

80C51-Based
8-Bit Microcontrollers

CONTENTS

80C51 Family Overview ... 3
- 8051 ... 3
- 80C51 ... 3
- 80CL51 ... 3
- 8052 ... 4
- 80C52 ... 4
- 83C053 ... 4
- 83CL410 ... 4
- 83C451 ... 4
- 83C528 ... 4
- 83C550 ... 4
- 83C552 ... 4
- 83C562 ... 6
- 83C575 ... 6
- 87C592 ... 6
- 83C652 ... 6
- 83C654 ... 6
- 83C751 ... 6
- 83C752 ... 6
- 83C851 ... 6
- 83C852 ... 6

80C51 Architecture ... 8
- Memory Organization ... 8
  - Program Memory ... 8
  - Data Memory ... 9
- 80C51 Family Instruction Set ... 11
  - Program Status Word ... 11
  - Addressing Modes ... 12
    - Direct Addressing ... 12
    - Indirect Addressing ... 12
    - Register Instructions ... 12
    - Register-Specific Instructions ... 12
    - Immediate Constants ... 12
    - Indexed Addressing ... 12
  - Arithmetic Instructions ... 12
  - Logical Instructions ... 13
  - Data Transfers ... 13
    - Internal RAM ... 13
    - External RAM ... 14
    - Lookup Tables ... 15
  - Boolean Instructions ... 15
    - Relative Offset ... 15
  - Jump Instructions ... 16
  - CPU Timing ... 17
  - Machine Cycles ... 17
  - Interrupt Structure ... 18
    - Interrupt Enables ... 18
    - Interrupt Priorities ... 22
    - Simulating a Third Priority Level in Software ... 22

Hardware Description ... 23
- Special Function Registers ... 24
  - Accumulator ... 24
  - B Register ... 24
  - Program StatusWord ... 24
  - Stack Pointer ... 24
  - Data Pointer ... 24
  - Ports 0 to 3 ... 24
  - Serial Data Buffer ... 24
  - Timer Registers Basic to 80C51 ... 24
  - Control Register for the 80C51 ... 24

80C51 Family Overview SECTION 1

CONTENTS (Continued)

Hardware Description (continued)
Port Structures and Operation ... 24
I/O Configurations ... 25
Writing to a Port ... 25
Port Loading and Interfacing ... 26
Read-Modify-Write Feature ... 27
Accessing External Memory ... 28
Timer/Counters ... 28
Timer 0 and Timer 1 ... 28
Mode 0 ... 28
Mode 1 ... 28
Mode 2 ... 28
Mode 3 ... 28
Standard Serial Interface ... 31
Multiprocessor Communications ... 31
Serial Port Control Register ... 31
Baud Rates ... 31
Using Timer 1 to Generate Baud Rates ... 31
More About Mode 0 ... 32
More About Mode 1 ... 33
More About Modes 2 and 3 ... 33
Interrupts ... 38
Priority Level Structure ... 38
How Interrupts Are Handled ... 39
External Interrupts ... 40
Response Time ... 40
Single-Step Operation ... 40
Reset ... 40
Power-on Reset ... 41
Power-Saving Modes of Operation ... 41
CMOS Power Reduction Mode ... 41
Idle Mode ... 41
Power-Down Mode ... 42
ONCE Mode ... 42
The On-Chip Oscillators ... 43
NMOS Version ... 43
CMOS Versions ... 43
Internal Timing ... 44
80C51 Pin Descriptions ... 44

Programmer's Guide and Instruction Set ... 48
Memory Organization ... 48
Program Memory ... 48
Direct and Indirect Address Area ... 48
80C51 Family Instruction Set ... 60
Instruction Definitions ... 64

EPROM Products ... 103
Programming the 87C51 ... 103
Encryption Table ... 103
Lock Bit ... 103
Program Verification ... 104
Signature Bytes ... 104
EPROM Erasure ... 105
Programming the 87C751 and 87C752 ... 105
Program Verification ... 106
87C751 and 87C752 Signature Bytes ... 106
Programming Features ... 106
Erasure Characteristics ... 106

8031AH/8051AH Data Sheet ... 107

80C31/80C51/87C51 Data Sheet ... 116

80C51 family overview

SECTION 1
80C51 FAMILY

80C51 FAMILY OVERVIEW

The Philips 80C51 family of products is based on the industry standard for 8-bit high performance microcontrollers. The architecture for the family has been optimized for sequential real time control applications. The 80C51 family of products are used in a wide range of applications from those that are relatively simple to applications in medical instrumentation and automobile control systems. All of the devices included in the family are available in versions that have either internal ROM, EPROM, or CPU only. With the exception of the 83C751 and 752 (which are limited to on-board memory) all of the devices in the family can address up to 64k bytes of both program and data memory.

The 80C51 family of microcontrollers includes the devices listed in Table 1. The basic architecture of these devices is shown in Figure 1.

8051

The 8051 is the original member of the family. Among the features of the 8051 are:

- 8-bit CPU optimized for control applications
- Extensive Boolean processing (single-bit logic) capabilities
- 32 bidirectional and individually addressable I/O lines
- 128 bytes of on-chip data RAM
- Two 16-bit timer/counters
- Full duplex UART
- 5-source interrupt structure with 2 priority levels
- On-chip clock oscillator
- 4k bytes of on-chip program memory
- 64k bytes program memory address space
- 64k bytes data memory address space
- 40-pin DIP and 44-pin PLCC packages

The 8031 is a CPU only version of the 8051 and differs from the 8051 in that it does not have on-chip ROM. The 8031 fetches all instructions from external memory.

80C51

The 80C51 is the CMOS version of the 8051. Functionally it is fully compatible with the 8051, but being CMOS it draws less current than its NMOS counterpart.

The ROMless version of the 80C51 is the 80C31. The EPROM version is the 87C51.

80CL51

The 80CL51 is a low power version of the 80C51. Functionally it is fully compatible with the 80C51 and 8051. The part can be operated at voltages from 1.8V to 6V and at oscillator frequencies from DC to 12MHz. The main benefit of this part is its ability to significantly reduce the current consumption in an application when it is operated at voltages and frequencies that are lower than those which the 80C51 will operate.

Table 1. 80C51 Family of Microcontrollers

DEVICE NAME	ROMless VERSION	EPROM VERSION	ROM BYTES	RAM BYTES	16-BIT TIMERS	CIRCUIT TYPE
8051	8031	–	4k	128	2	NMOS
80C51	80C31	87C51	4k	128	2	HMOS
80CL51	–	–	4k	128	2	SACMOS
8052	8032	–	8k	256	3	NMOS
80C52	80C32	87C52	8k	256	3	CMOS
83C053	–	87C054	8k	192	2	CMOS
83CL410	80CL410	–	4k	128	2	CMOS
83C451	80C451	87C451	4k	128	2	CMOS
83C528	80C528	87C528	32k	512	3 + WD	CMOS
83C550	80C550	87C550	4k	128	2 + WD	CMOS
83C552	80C552	87C552	8k	256	3 + WD	CMOS
83C562	80C526	–	8k	256	3 + WD	CMOS
87C575	80C575	87C575	8k	256	3 + PCA + WD	CMOS
87C592	–	87C592	16k	512	3 + WD	CMOS
83C652	80C652	87C652	8k	256	2	CMOS
83C654	–	87C654	16k	256	2	CMOS
83C751	–	87C751	2k	64	1	CMOS
83C752	–	87C752	2k	64	1	CMOS
83C851	80C851	–	4k	128	2	CMOS
83C852	–	–	6k	256	2	CMOS

8052

The 8052 is an enhanced version of the 8051. It is fabricated with NMOS technology, and is upwards compatible with the 8051. Its enhancements over the 8051 include:

- 256 bytes of on-chip data RAM
- Three counter/timers
- A 6-source interrupt structure
- 8k bytes of on-chip program memory
- 40-pin DIP and 44-pin PLCC packages

The ROMless version of the 8052 is the 8032.

80C52

The 80C52 is the CMOS version of the 8052. Functionally it is fully compatible with the 8052, but being CMOS it draws less current than its NMOS counterpart.

The ROMless version of the 80C52 is the 80C32. The EPROM version is the 87C52.

83C053

The 83C053 is a derivative of the 80C51 that is intended for use as the central control mechanism in a television. It provides tuner functions and has an on-screen display facility. The main features of the part are:

- 8k ROM (83C053)
- 16k EPROM (87C054)
- 192 bytes RAM
- On-screen display controller
- Three digital video outputs
- Multiplexer/mixer and background intensity controls
- 128 × 10 display RAM
- 60 × 18 × 14 character generator ROM
- Eight 6-bit PWM
- One 14-bit PWM
- Four high current open-drain port outputs
- Twelve high voltage (+12V) open-drain outputs
- Programmable video input and output polarities
- 42-pin shrink DIP

There is no ROMless version of the part. The EPROM version is the 87C054.

83CL410

The 83CL410 is a derivative of the 80C51 that operates at very low supply levels and frequency. At lower supply levels and frequency, the part has dramatically reduced power dissipation. This part is ideally suited for low voltage battery operation. The part has all of the features of an 80C51, except the full-duplex UART. The additional features of the 83CL410 are:

- I^2C serial interface
- Warm start from power-down mode
- 32kHz to 20MHz frequency operation (can be operated to DC with an external oscillator)
- Power supply range: 1.5 to 6V

This part is socket compatible with the 80C51. The ROMless version is the 80CL410. There is no EPROM version.

83C451

The 83C451 is an extended I/O version of the 80C51 with the following features:

- Seven 8-bit quasi-bidirectional I/O ports (PLCC version)
- Six 8-bit and one 4-bit quasi-bidirectional I/O ports (DIP version)
- Mailbox port (port 6) features:
  - Operation as normal quasi-bidirectional I/O port
  - Four handshake control pins
  - Control status register
  - Input and output buffer registers making port 6 suitable for:
    direct MPU interface
    parallel printer interface
- 64-pin DIP and 68-pin PLCC packages

All other aspects of the 83C451 are identical to the 80C51. The ROMless version of the 83C451 is the 80C451, and the 87C451 is the EPROM version.

83C528

This is an extended memory version of the 80C51. It is socket compatible with the 80C51 and has all of the 80C51 features plus:

- 32k bytes of ROM
- 512 bytes of RAM
- A third 16-bit counter/timer (identical to the 80C52 T2)
- A watchdog timer with separate oscillator
- An I^2C serial port
- Warm start from power-down mode

The ROMless version of the 83C528 is the 80C528, and the EPROM version is the 87C528.

83C550

The 83C550 is a 40-pin derivative of the 80C51 that has an 8 channel, 8-bit A/D converter. In addition to having all of the features of an 80C51, the part has:

- 8 channel, 8-bit A/D
- Watchdog timer

Although the 83C550 is available in a 40-pin package, it is not socket compatible with the 80C51, because of its analog features.

The ROMless version of the 83C550 is the 80C550 and the EPROM version is the 87C550.

83C552

The 83C552 is an extended function 80C51 with the following features:

- 8k bytes of ROM
- 256 bytes RAM
- 10-bit 8 channel A/D
- Counter/timer array with high speed outputs and capture inputs
- Four counter/timers (including a watchdog timer)
- Two PWM outputs
- 8051 full duplex UART
- Six 8-bit I/O ports
- I^2C serial port
- 68-pin PLCC package

The 83C552 is 100% code compatible with the 80C51. The ROMless version of the 83C552 is the 80C552 and the EPROM version is the 87C552.

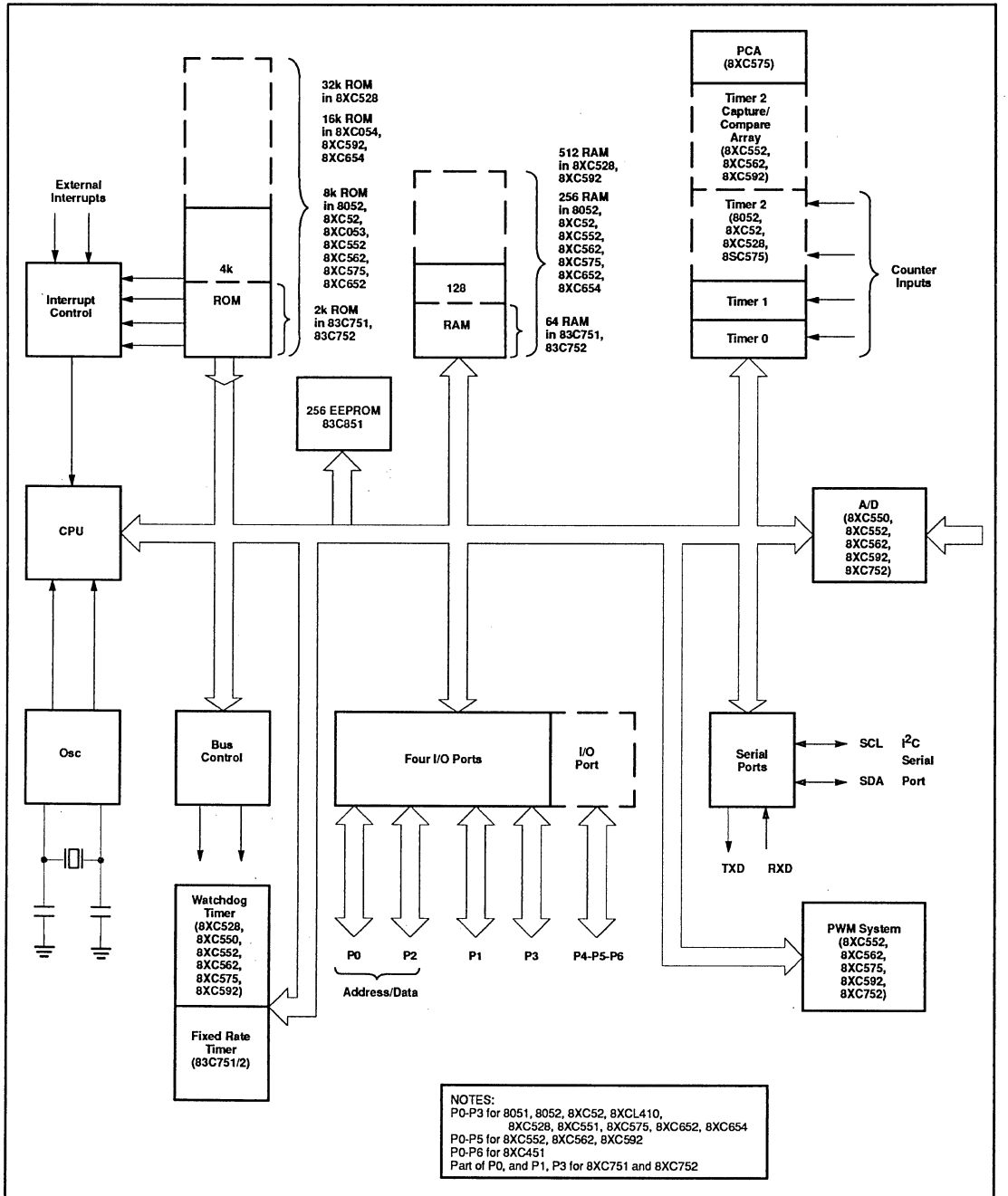


Figure 1. 80C51 Family Block Diagram

83C562

The 83C562 is an 80C51 derivative that is identical to the 83C552 except that the I²C interface has been removed and the A/D converter is 8 bit instead of 10 bit.

The ROMless version of the 83C562 is the 80C562. There is no EPROM version. For development and prototyping, the 87C552 can be used.

83C575

The 83C575 is an 80C51 derivative that contains all of the best derivative features that are available today. It is fully code compatible with the 80C51 and has the following features:

- 8k program memory
- 256 bytes RAM
- 3 16-bit timers
- Programmable counter array
- Watchdog timer
- Oscillator fail detection
- Power fail detection
- Enhanced UART
- Power On flag
- Low active reset
- Port pins asynchronously reset low
- 4 analog comparators
- Port 2 active pull-ups can be disabled for open drain operation
- 4 8-bit I/O ports
- 40-pin DIP, 44-pin PLCC, 44-pin QFP

The ROMless version of the 83C575 is the 80C575 and the EPROM version is the 87C575. The EPROM version is available in both UV erasable and OTP.

87C592

The 87C592 is a microcontroller that is functionally fully compatible with the 80C51 and it has a Control Area Network (CAN) bus interface on-chip. The 87C592 has all of the features of the 83C552 with the exception of the I²C serial interface. In addition, the 87C592 has the following:

- CAN bus interface
- 16k EPROM program memory
- 512 bytes RAM
- DMA transfer between the on-chip RAM to the CAN interface

The 87C592 is available in 68-pin PLCC. Both UV erasable and OTP versions are available.

83C652

The 83C652 is an 80C51 with the following additions: an 8k ROM, 256 bytes RAM and I²C serial port.

The 83C652 is pin-for-pin compatible with the 80C51 except for minor DC specifications on the I²C serial port pins. The ROMless version of the 83C652 is the 80C652 and the EPROM version is the 87C652.

83C654

This 80C51 derivative is identical to the 83C652 except that it has 16k of program ROM. It is pin-for-pin socket compatible with the 80C51.

There is no ROMless version of this part because it would be identical to an 80C652. The EPROM version is the 87C654.

83C751

The 83C751 is a 24-pin derivative of the 80C51, for applications where small size and cost are of prime consideration. The 83C751 is packaged in a 24-pin "skinny-dip" (.300 wide) and in a 28-pin PLCC package. The following differences exist between the 83C751 and the 80C51. The 83C751 has:

- 2k bytes ROM
- 64 bytes RAM
- I²C serial port (no UART)
- 19 I/O lines
- Single level interrupt structure
- One counter/timer with 16-bit autoload
- No external memory expandability (data memory can be expanded using the I²C serial port and I²C compatible memory devices)

Note that since there is no external expandability, the external memory addressing signals: $\overline{WR}$, $\overline{RD}$, $\overline{PSEN}$, $\overline{EA}$, and ALE are not present. Because of these differences, the instructions LJMP, LCALL, and MOVX execute as NOPs in the 83C751.

For all other instructions the 83C751 is 100% code compatible with the 80C51 and operates at full 80C51 speed. The EPROM version of the 83C751 is the 87C751. There is no ROMless version.

83C752

The 83C752 is a 28-pin derivative of the 80C51 that is intended for use in automotive, electro-mechanical, and consumer applications. The 83C752 contains most of the features of the 80C51 with the following differences:

- 2k bytes ROM
- 64 bytes RAM
- Single level interrupt structure
- One 16-bit counter/timer with 16-bit autoload
- Two 8-bit and one 5-bit bidirectional I/O ports
- I²C serial interface
- One PWM with timer, including overflow interrupt capability
- Five channels of 8-bit A/D
- 28-pin packages, both DIP and PLCC

The 83C752 does not have external memory expandability. The EPROM version of the 83C752 is the 87C752. There is no ROMless version.

83C851

This 80C51 derivative has on-chip EEPROM. It is socket compatible with the 80C51 and has all of the features of the 80C51. In addition to these standard features it has:

- 256 bytes of EEPROM
- EEPROM security mode
- ROM code protection

The ROMless version of the part is the 80C851. There is no EPROM version.

83C852

The 83C852 is an 80C51 derivative that has been developed for Secure Smart Card applications. The microcontroller has the same instruction set as the 80C51. It has been specifically designed for conditional access and provides the highest level of software and access security. The features of the part are as follows:

- 6k ROM program memory
- 256 bytes RAM
- 2k bytes of EEPROM
- Security features
- Two 16-bit timers
- A special calculation unit that speeds up the execution time of public keys and secret keys cryptographic algorithms
- Low frequency detect
- Two I/O lines

The 83C852 is a 6-pin microcontroller that is available in ROM version only.

Table 2. 80C51 Derivative Comparisons

DEVICE	A/D	PORTS	PWM	TIMERS	SERIAL PORT
80C51	–	4	–	Two standard	UART
80CL51	–	4	–	Two standard	UART
80C52	–	4	–	Two standard, timer 2	UART
83C053	–	$3^{1}/_{2}$	9	Two standard	–
83CL410	–	4	–	Two standard	I^2C
83C451	–	7	–	Two standard	UART
83C528	–	4	–	Two standard, timer 2, watchdog (4 total)	UART, I^2C
83C550	8 channel/8-bit	4	–	Two standard, watchdog	UART
83C552	8 channel/10-bit	6	2	Two standard, timer 2, watchdog (4 total)	UART, I^2C
83C562	8 channel/8-bit	6	2	Two standard, timer 2, watchdog (4 total)	UART
83C575	4 comparators	4	5	Two standard, timer 2, PCA, watchdog	UART
87C592	8 channel/10-bit	6	2	Two standart, timer 2, watchdog	UART, CAN
83C652	–	4	–	Two standard	UART, I^2C
83C654	–	4	–	Two standard	UART, I^2C
83C751	–	$2^{3}/_{8}$	–	One standard, extended to 16-bit autoload	I^2C
83C752	5 channel/8-bit	$2^{5}/_{8}$	1	One standard, extended to 16-bit autoload	I^2C
83C851	–	4	–	Two standard	UART
83C852	–	$^{2}/_{8}$	–	Two standard	–

80C51 ARCHITECTURE

MEMORY ORGANIZATION

All 80C51 devices have separate address spaces for program and data memory, as shown in Figures 1 and 2. The logical separation of program and data memory allows the data memory to be accessed by 8-bit addresses, which can be quickly stored and manipulated by an 8-bit CPU. Nevertheless, 16-bit data memory addresses can also be generated through the DPTR register.

Program memory (ROM, EPROM) can only be read, not written to. There can be up to 64k bytes of program memory. In the 80C51, the lowest 4k bytes of program are on-chip. In the ROMless versions, all program memory is external. The read strobe for external program memory is the $\overline{PSEN}$ (program store enable).

Data Memory (RAM) occupies a separate address space from Program Memory. In the 80C51, the lowest 128 bytes of data memory are on-chip. Up to 64k bytes of external RAM can be addressed in the external Data Memory space. In the ROMless version, the lowest 128 bytes are on-chip. The CPU generates read and write signals, $\overline{RD}$ and $\overline{WR}$, as needed during external Data Memory accesses.

External Program Memory and external Data Memory may be combined if desired by applying the RD and PSEN signals to the inputs of an AND gate and using the output of the gate as the read strobe to the external Program/Data memory.

Program Memory

Figure 3 shows a map of the lower part of the Program Memory. After reset, the CPU begins execution from location 0000H. As shown in Figure 3, each interrupt is assigned a fixed location in Program Memory. The interrupt causes the CPU to jump to that location, where it commences execution of the service routine. External Interrupt 0, for example, is assigned to location 0003H. If External Interrupt 0 is going to be used, its service routine must begin at location 0003H. If the interrupt is not going to be used, its service location is available as general purpose Program Memory.

The interrupt service locations are spaced at 8-byte intervals: 0003H for External Interrupt 0, 000BH for Timer 0, 0013H for External Interrupt 1, 001BH for Timer 1, etc. If an interrupt service routine is short enough (as is often the case in control applications), it can reside entirely within that 8-byte interval. Longer service routines can use a jump instruction to skip over subsequent interrupt locations, if other interrupts are in use.

The lowest 4k bytes of Program Memory can either be in the on-chip ROM or in an external ROM. This selection is made by strapping the $\overline{EA}$ (External Access) pin to either V_{CC}, or V_{SS}. In the 80C51, if the $\overline{EA}$ pin is strapped to V_{CC}, then the program fetches to addresses 0000H through 0FFFH are directed to the internal ROM. Program fetches to addresses 1000H through FFFFH are directed to external ROM.

If the $\overline{EA}$ pin is strapped to V_{SS}, then all program fetches are directed to external ROM. The ROMless parts (8031, 80C31, etc.) must have this pin externally strapped to V_{SS} to enable them to execute from external Program Memory.

The read strobe to external ROM, $\overline{PSEN}$, is used for all external program fetches. $\overline{PSEN}$ is not activated for internal program fetches.

The hardware configuration for external program execution is shown in Figure 4. Note that 16 I/O lines (Ports 0 and 2) are dedicated to bus functions during external Program Memory fetches. Port 0 (P0 in Figure 4) serves as a multiplexed address/data bus. It emits the low byte of the Program Counter (PCL) as an address, and then goes into a float state awaiting the arrival of the code byte from the Program Memory. During the time that the low byte of the Program Counter is valid on Port 0, the signal ALE (Address Latch Enable) clocks this byte into an address latch. Meanwhile, Port 2 (P2 in Figure 4) emits the high byte of the Program Counter (PCH). Then $\overline{PSEN}$ strobes the EPROM and the code byte is read into the microcontroller.

Program Memory addresses are always 16 bits wide, even though the actual amount of Program Memory used may be less than 64k bytes. External program execution sacrifices two of the 8-bit ports, P0 and P2, to the function of addressing the Program Memory.

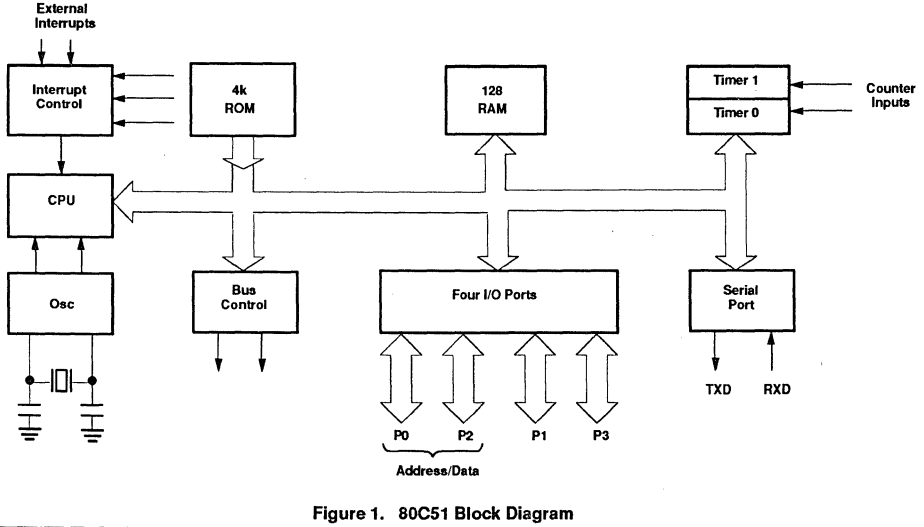


Figure 1. 80C51 Block Diagram

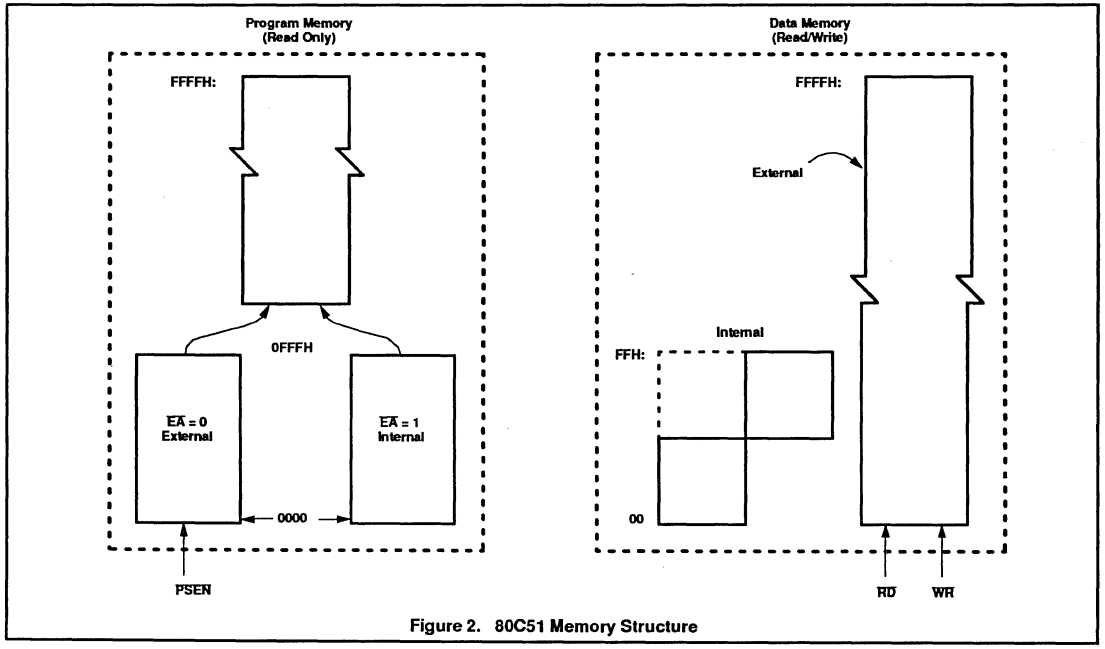


Figure 2. 80C51 Memory Structure

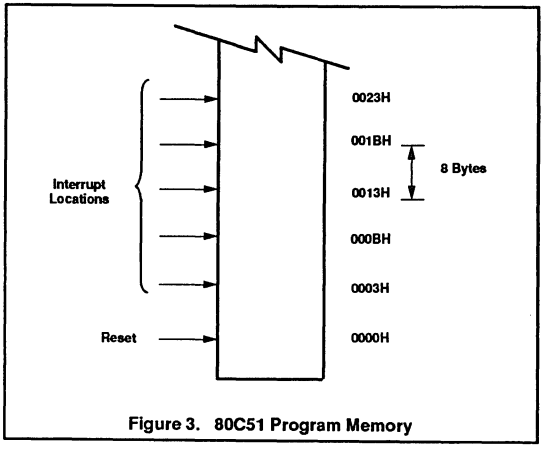


Figure 3. 80C51 Program Memory

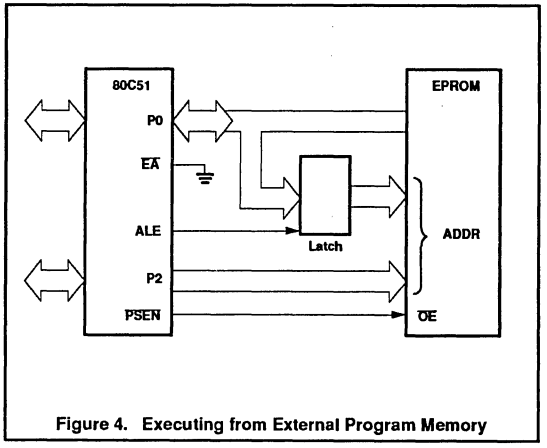


Figure 4. Executing from External Program Memory

Data Memory

The right half of Figure 2 shows the internal and external Data Memory spaces available to the 80C51 user. Figure 5 shows a hardware configuration for accessing up to 2k bytes of external RAM. The CPU in this case is executing from internal ROM. Port 0 serves as a multiplexed address/data bus to the RAM, and 3 lines of Port 2 are being used to page the RAM. The CPU generates $\overline{RD}$ and $\overline{WR}$ signals as needed during external RAM accesses. There can be up to 64k bytes of external Data Memory. External Data Memory addresses can be either 1 or 2 bytes wide. One-byte addresses are often used in conjunction with one or more other I/O lines to page the RAM, as shown in Figure 5.

Two-byte addresses can also be used, in which case the high address byte is emitted at Port 2.

Internal Data Memory is mapped in Figure 6. The memory space is shown divided into three blocks, which are generally referred to as the Lower 128, the Upper 128, and SFR space.

Internal Data Memory addresses are always one byte wide, which implies an address space of only 256 bytes. However, the addressing modes for internal RAM can in

fact accommodate 384 bytes, using a simple trick. Direct addresses higher than 7FH access one memory space, and indirect addresses higher than 7FH0 access a different memory space. Thus Figure 6 shows the Upper 128 and SFR space occupying the same block of addresses, 80H through FFH, although they are physically separate entities.

The Lower 128 bytes of RAM are present in all 80C51 devices as mapped in Figure 7. The lowest 32 bytes are grouped into 4 banks of 8 registers. Program instructions call out these registers as R0 through R7. Two bits in the Program Status Word (PSW) select which register bank is in use. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing.

The next 16 bytes above the register banks form a block of bit-addressable memory space. The 80C51 instruction set includes a wide selection of single-bit instructions, and the 128 bits in this area can be directly addressed by these instructions. The bit addresses in this area are 00H through 7FH.

All of the bytes in the Lower 128 can be accessed by either direct or indirect addressing. The Upper 128 (Figure 8) can only be accessed by indirect addressing.

Figure 9 gives a brief look at the Special Function Register (SFR) space. SFRs include the Port latches, timers, peripheral controls, etc. These registers can only be accessed by direct addressing. Sixteen addresses in SFR space are both byte- and bit-addressable. The bit-addressable SFRs are those whose address ends in 0H or 8H.

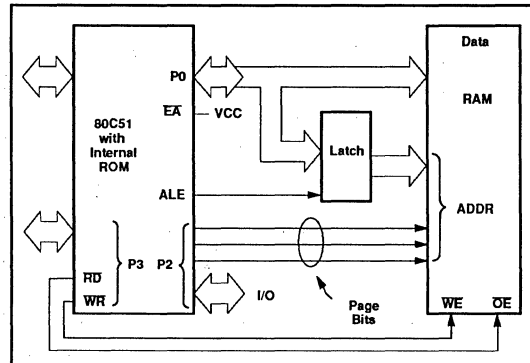


Figure 5. Accessing External Data Memory If the Program Memory Is Internal, the Other Bits of P2 Are Available as I/O

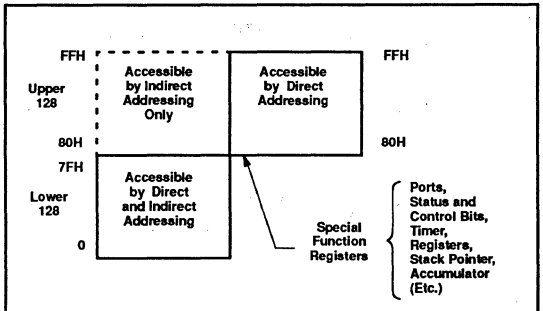


Figure 6. Internal Data Memory

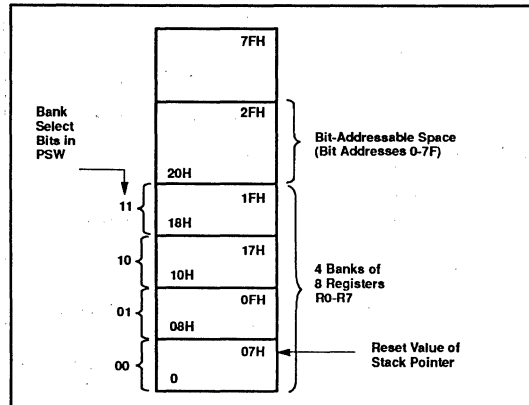


Figure 7. Lower 128 Bytes of Internal RAM

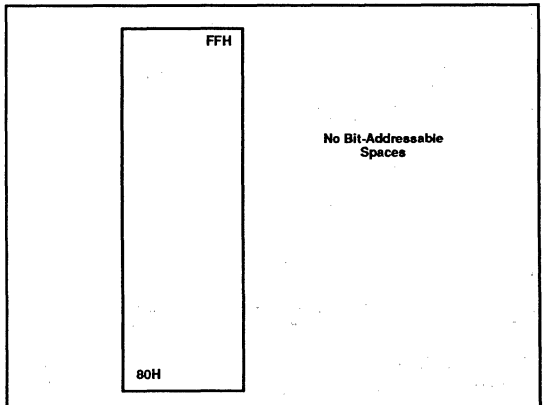


Figure 8. Upper 128 Bytes of Internal RAM

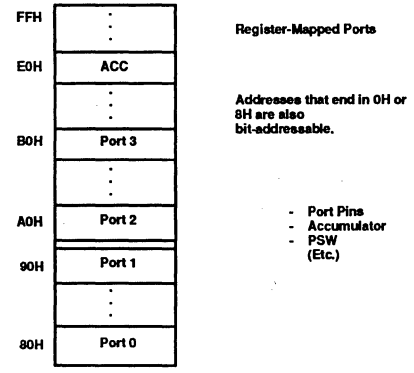


Figure 9. SFR Space

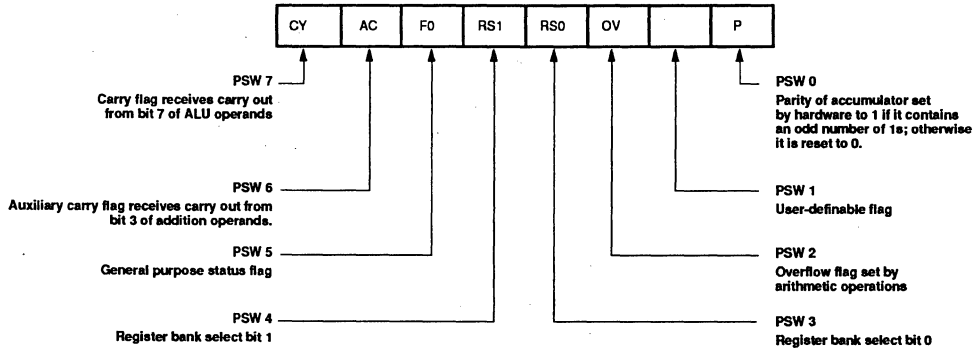


Figure 10. PSW (Program Status Word) Register in 80C51 Devices

80C51 FAMILY INSTRUCTION SET

The 80C51 instruction set is optimized for 8-bit control applications. It provides a variety of fast addressing modes for accessing the internal RAM to facilitate byte operations on small data structures. The instruction set provides extensive support for one-bit variables as a separate data type, allowing direct bit manipulation in control and logic systems that require Boolean processing.

Program Status Word

The Program Status Word (PSW) contains several status bits that reflect the current state of the CPU. The PSW, shown in Figure 10, resides in the SFR space. It contains the Carry bit, the Auxiliary Carry (for BCD operations), the two register bank select bits, the Overflow flag, a Parity bit, and two user-definable status flags.

The Carry bit, other than serving the function of a Carry bit in arithmetic operations, also serves as the "Accumulator" for a number of Boolean operations.

The bits RS0 and RS1 are used to select one of the four register banks shown in Figure 7. A number of instructions refer to these RAM locations as R0 through R7. The selection of which of the four is being referred to is made on the basis of the RS0 and RS1 at execution time.

The Parity bit reflects the number of 1s in the Accumulator: P = 1 if the Accumulator contains an odd number of 1s, and P = 0 if the Accumulator contains an even number of 1s. Thus the number of 1s in the Accumulator plus P is always even. Two bits in the PSW are uncommitted and may be used as general purpose status flags.

Addressing Modes

The addressing modes in the 80C51 instruction set are as follows:

Direct Addressing

In direct addressing the operand is specified by an 8-bit address field in the instruction. Only internal Data RAM and SFRs can be directly addressed.

Indirect Addressing

In indirect addressing the instruction specifies a register which contains the address of the operand. Both internal and external RAM can be indirectly addressed.

The address register for 8-bit addresses can be R0 or R1 of the selected bank, or the Stack Pointer. The address register for 16-bit addresses can only be the 16-bit "data pointer" register, DPTR.

Register Instructions

The register banks, containing registers R0 through R7, can be accessed by certain instructions which carry a 3-bit register specification within the opcode of the instruction. Instructions that access the registers this way are code efficient, since this mode eliminates an address byte. When the instruction is executed, one of the eight registers in the selected bank is accessed. One of four banks is selected at execution time by the two bank select bits in the PSW.

Register-Specific Instructions

Some instructions are specific to a certain register. For example, some instructions always operate on the Accumulator, or Data Pointer, etc., so no address byte is needed to point to it. The opcode itself does that. Instructions that refer to the Accumulator as A assemble as accumulator specific opcodes.

Immediate Constants

The value of a constant can follow the opcode in Program Memory. For example,

```
MOV  A, #100
```

loads the Accumulator with the decimal number 100. The same number could be specified in hex digits as 64H.

Indexed Addressing

Only program Memory can be accessed with indexed addressing, and it can only be read. This addressing mode is intended for reading look-up tables in Program Memory A 16-bit base register (either DPTR or the Program Counter) points to the base of the table, and the Accumulator is set up with the table entry number.

The address of the table entry in Program Memory is formed by adding the Accumulator data to the base pointer.

Another type of indexed addressing is used in the "case jump" instruction. In this case the destination address of a jump instruction is computed as the sum of the base pointer and the Accumulator data.

Arithmetic Instructions

The menu of arithmetic instructions is listed in Table 1. The table indicates the addressing modes that can be used with each instruction to access the <byte> operand. For example, the ADD A,<byte> instruction can be written as:

```
ADD   a, 7FH (direct addressing)
ADD   A, @R0 (indirect addressing)
ADD   a, R7 (register addressing)
ADD   A, #127 (immediate constant)
```

The execution times listed in Table 1 assume a 12MHz clock frequency. All of the arithmetic instructions execute in 1µs except the INC DPTR instruction, which takes 2µs, and the Multiply and Divide instructions, which take 4µs.

Note that any byte in the internal Data Memory space can be incremented without going through the Accumulator.

One of the INC instructions operates on the 16-bit Data Pointer. The Data Pointer is used to generate 16-bit addresses for external memory, so being able to increment it in one 16-bit operation is a useful feature.

The MUL AB instruction multiplies the Accumulator by the data in the B register and puts the 16-bit product into the concatenated B and Accumulator registers.

The DIV AB instruction divides the Accumulator by the data in the B register and leaves the 8-bit quotient in the Accumulator, and the 8-bit remainder in the B register.

Oddly enough, DIV AB finds less use in arithmetic "divide" routines than in radix conversions and programmable shift operations. An example of the use of DIV AB in a radix conversion will be given later. In shift operations, dividing a number by 2n shifts its n bits to the right. Using DIV AB to perform the division completes the shift in 4µs and leaves the B register holding the bits that were shifted out. The DA A instruction is for BCD arithmetic operations. In BCD arithmetic, ADD and ADDC instructions should always be followed by a DA A operation, to ensure that the result is also in BCD. Note that DA A will not convert a binary number to BCD. The DA A operation produces a meaningful result only as the second step in the addition of two BCD bytes.

Table 1. 80C51 Arithmetic Instructions

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION
		DIR	IND	REG	IMM	TIME (µs)
ADD A,<byte>	A = A + <byte>	X	X	X	X	1
ADDC A,<byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A,<byte>	A = A – <byte> – C	X	X	X	X	1
INC A	A = A + 1	Accumulator only				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Data Pointer only				2
DEC A	A = A – 1	Accumulator only				1
DEC <byte>	<byte> = <byte> – 1	X	X	X		1
MUL AB	B:A = B x A	ACC and B only				4
DIV AB	A = Int[A/B] B = Mod[A/B]	ACC and B only				4
DA A	Decimal Adjust	Accumulator only				1

Logical Instructions

Table 2 shows the list of 80C51 logical instructions. The instructions that perform Boolean operations (AND, OR, Exclusive OR, NOT) on bytes perform the operation on a bit-by-bit basis. That is, if the Accumulator contains 00110101B and byte contains 01010011B, then:

```
ANL   A, <byte>
```

will leave the Accumulator holding 00010001B.

The addressing modes that can be used to access the <byte> operand are listed in Table 2.

The ANL A, <byte> instruction may take any of the forms:

```
ANL   A,7FH (direct addressing)
ANL   A,@R1 (indirect addressing)
ANL   A,R6 (register addressing)
ANL   A,#53H (immediate constant)
```

All of the logical instructions that are Accumulator-specific execute in 1µs (using a 12MHz clock). The others take 2µs.

Note that Boolean operations can be performed on any byte in the internal Data Memory space without going through the Accumulator. The XRL <byte>, #data instruction, for example, offers a quick and easy way to invert port bits, as in XRL P1, #OFFH.

If the operation is in response to an interrupt, not using the Accumulator saves the time and effort to push it onto the stack in the service routine.

The Rotate instructions (RL, A, RLC A, etc.) shift the Accumulator 1 bit to the left or right. For a left rotation, the MSB rolls into the LSB position. For a right rotation, the LSB rolls into the MSB position.

The SWAP A instruction interchanges the high and low nibbles within the Accumulator. This is a useful operation in BCD manipulations. For example, if the Accumulator contains a binary number which is known to be less than 100, it can be quickly converted to BCD by the following code:

```
MOVE  B,#10
DIV   AB
SWAP  A
ADD   A,B
```

Dividing the number by 10 leaves the tens digit in the low nibble of the Accumulator, and the ones digit in the B register. The SWAP and ADD instructions move the tens digit to the high nibble of the Accumulator, and the ones digit to the low nibble.

Data Transfers

Internal RAM

Table 3 shows the menu of instructions that are available for moving data around within the internal memory spaces, and the addressing modes that can be used with each one. With a 12MHz clock, all of these instructions execute in either 1 or 2µs.

The MOV <dest>, <src> instruction allows data to be transferred between any two internal RAM or SFR locations without going through the Accumulator. Remember, the Upper 128 bytes of data RAM can be accessed only by indirect addressing, and SFR space only by direct addressing.

Note that in 80C51 devices, the stack resides in on-chip RAM, and grows upwards. The PUSH instruction first increments the Stack Pointer (SP), then copies the byte into the stack. PUSH and POP use only direct addressing to identify the byte being saved or restored, but the stack itself is accessed by indirect addressing using the SP register. This means the stack can go into the Upper 128 bytes of RAM, if they are implemented, but not into SFR space.

The Upper 128 bytes of RAM are not implemented in the 80C51 nor in its ROMless or EPROM counterparts. With these devices, if the SP points to the Upper 128, PUSHed bytes are lost, and POPed bytes are indeterminate.

The Data Transfer instructions include a 16-bit MOV that can be used to initialize the Data Pointer (DPTR) for look-up tables in Program Memory, or for 16-bit external Data Memory accesses.

Table 2. 80C51 Logical Instructions

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION
		DIR	IND	REG	IMM	TIME (µs)
ANL A,<byte>	A = A.AND. <byte>	X	X	X	X	1
ANL <byte>,A	<byte> = <byte> .AND.A	X				1
ANL <byte>,#data	<byte> = <byte> .AND.#data	X				2
ORL A,<byte>	A = A.OR.<byte>	X	X	X	X	1
ORL <byte>,A	<byte> = <byte> .OR.A	X				1
ORL <byte>,#data	<byte> = <byte> .OR.#data	X				2
XRL A,<byte>	A = A.XOR. <byte>	X	X	X	X	1
XRL <byte>,A	<byte> = <byte> .XOR.A	X				1
XRL <byte>,#data	<byte> = <byte> .XOR.#data	X				2
CRL A	A = 00H	Accumulator only				1
CPL A	A = .NOT.A	Accumulator only				1
RL A	Rotate ACC Left 1 bit	Accumulator only				1
RLC A	Rotate Left through Carry	Accumulator only				1
RR A	Rotate ACC Right 1 bit	Accumulator only				1
RRC A	Rotate Right through Carry	Accumulator only				1
SWAP A	Swap Nibbles in A	Accumulator only				1

Table 3. Data Transfer Instructions that Access Internal Data Memory Space

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION
		DIR	IND	REG	IMM	TIME (µs)
MOV A,<src>	A = <src>	X	X	X	X	1
MOV <dest>,A	<dest> = A	X	X	X		1
MOV <dest>,<src>	<dest> = <src>	X	X	X	X	2
MOV DPTR,#data16	DPTR = 16-bit immediate constant				X	2
PUSH <src>	INC SP:MOV"@SP",<src>	X				2
POP <dest>	MOV <dest>,"@SP":DEC SP	X				2
XCH A,<byte>	ACC and <byte> exchange data	X	X	X		1
XCHD A,@Ri	ACC and @Ri exchange low nibbles		X			1

The XCH A, <byte> instruction causes the Accumulator and addressed byte to exchange data. The XCHD A, @Ri instruction is similar, but only the low nibbles are involved in the exchange.

To see how XCH and XCHD can be used to facilitate data manipulations, consider first the problem of shifting an 8-digit BCD number two digits to the right. Figure 11 shows how this can be done using direct MOVs, and for comparison how it can be done using XCH instructions. To aid in understanding how the code works, the contents of the registers that are holding the BCD number and the content of the Accumulator are shown alongside each instruction to indicate their status after the instruction has been executed.

After the routine has been executed, the Accumulator contains the two digits that were shifted out on the right. Doing the routine with direct MOVs uses 14 code bytes and 9µs of execution time (assuming a 12MHz clock). The same operation with XCHs uses only 9 bytes and executes almost twice as fast.

To right-shift by an odd number of digits, a one-digit shift must be executed.

Figure 12 shows a sample of code that will right-shift a BCD number one digit, using the XCHD instruction. Again, the contents of the registers holding the number and of the Accumulator are shown alongside each instruction.

First, pointers R1 and R0 are set up to point to the two bytes containing the last four BCD digits. Then a loop is executed which leaves the last byte, location 2EH, holding the last two digits of the shifted number. The pointers are decremented, and the loop is repeated for location 2DH. The CJNE instruction (Compare and Jump if Not Equal) is a loop control that will be described later. The loop executed from LOOP to CJNE for R1 = 2EH, 2DH, 2CH, and 2BH. At that point the digit that was originally shifted out on the right has propagated to location 2AH. Since that location should be left with 0s, the lost digit is moved to the Accumulator.

External RAM

Table 4 shows a list of the Data Transfer instructions that access external Data Memory. Only indirect addressing can be used. The choice is whether to use a one-byte address, @Ri, where Ri can be either R0 or R1 of the selected register bank, or a two-byte address, @DPTR. The disadvantage to using 16-bit addresses if only a few k bytes of external RAM are involved is that 16-bit addresses use all 8 bits of Port 2 as address bus. On the other hand, 8-bit addresses allow one to address a few bytes of RAM, as shown in Figure 5, without having to sacrifice all of Port 2. All of these instructions execute in 2 µs, with a 12MHz clock.

Note that in all external Data RAM accesses, the Accumulator is always either the destination or source of the data.

The read and write strobes to external RAM are activated only during the execution of a MOVX instruction. Normally these signals are inactive, and in fact if they're not going to be used at all, their pins are available as extra I/O lines.

		2A	2B	2C	2D	2E	ACC
MOV	A,2EH	00	12	34	56	78	78
MOV	2EH,2DH	00	12	34	56	56	78
MOV	2DH,2CH	00	12	34	34	56	78
MOV	2CH,2BH	00	12	12	34	56	78
MOV	2BH,#0	00	00	12	34	56	78

A. Using direct MOVs: 14 bytes, 9 µs

		2A	2B	2C	2D	2E	ACC
CLR	A	00	12	34	56	78	00
XCH	A,2BH	00	00	34	56	78	12
XCH	A,2CH	00	00	12	56	78	34
XCH	A,2DH	00	00	12	34	78	56
XCH	A2EH	00	00	12	34	56	78

B. Using XCHs: 9 bytes, 5 µs

Figure 11. Shifting a BCD Number Two Digits to the Right

			2A	2B	2C	2D	2E	ACC
	MOV	R1,#2EH	00	12	34	56	78	XX
	MOV	R0,#2DH	00	12	34	56	78	XX
	loop for R1 = 2EH:							
LOOP:	MOV	A,@R1	00	12	34	56	78	78
	XCHD	A,@R0	00	12	34	58	78	76
	SWAP	A	00	12	34	58	78	67
	MOV	@R1,A	00	12	34	58	67	67
	DEC	R1	00	12	34	58	67	67
	DEC	R0	00	12	34	58	67	67
	CJNE	R1,#2AH,LOOP						
	loop for R1 = 2DH:		00	12	38	45	67	45
	loop for R1 = 2CH:		00	18	23	45	67	23
	loop for R1 = 2BH:		08	01	23	45	67	01
	CLR	A	08	01	23	45	67	00
	XCH	A,2AH	00	01	23	45	67	08

Figure 12. Shifting a BCD Number One Digit to the Right

Table 4. 80C51 Data Transfer Instructions that Access External Data Memory Space

ADDRESS WIDTH	MNEMONIC	OPERATION	EXECUTION TIME (μs)
8 bits	MOVX A,@Ri	Read external RAM @Ri	2
8 bits	MOVX @Ri,A	Write external RAM @ Ri	2
16 bits	MOVX A,@DPTR	Read external RAM @ DPTR	2
16 bits	MOVX @DPTR,A	Write external RAM @ DPTR	2

Lookup Tables

Table 5 shows the two instructions that are available for reading lookup tables in Program Memory. Since these instructions access only Program Memory, the lookup tables can only be read, not updated.

If the table access is to external Program Memory, then the read strobe is $\overline{PSEN}$.

The mnemonic is MOVC for "move constant." The first MOVC instruction in Table 5 can accommodate a table of up to 256 entries numbered 0 through 255. The number of the desired entry is loaded into the Accumulator, and the Data Pointer is set up to point to the beginning of the table. Then:

```
MOVC  A,@A+DPTR
```

copies the desired table entry into the Accumulator.

The other MOVC instruction works the same way, except the Program Counter (PC) is used as the table base, and the table is accessed through a subroutine. First the number of the desired entry is loaded into the Accumulator, and the subroutine is called:

```
MOV   A,ENTRY NUMBER
CALL  TABLE
```

The subroutine "TABLE" would look like this:

```
TABLE: MOVC    A,@A+PC
       RET
```

The table itself immediately follows the RET (return) instruction in Program Memory. This type of table can have up to 255 entries, numbered 1 through 255. Number 0 cannot be used, because at the time the MOVC instruction is executed, the PC contains the address of the RET instruction. An entry numbered 0 would be the RET opcode itself.

Boolean Instructions

80C51 devices contain a complete Boolean (single-bit) processor. The internal RAM contains 128 addressable bits, and the SFR space can support up to 128 addressable bits as well. All of the port lines are bit-addressable, and each one can be treated as a separate single-bit port. The instructions that access these bits are not just conditional branches, but a complete menu of move, set, clear, complement, OR, and AND instructions. These kinds of bit operations are not easily obtained in other architectures with any amount of byte-oriented software.

The instruction set for the Boolean processor is shown in Table 6. All bit accesses are by direct addressing.

Bit addresses 00H through 7FH are in the Lower 128, and bit addresses 80H through FFH are in SFR space.

Note how easily an internal flag can be moved to a port pin:

```
MOV   C,FLAG
MOV   P1.0,C
```

In this example, FLAG is the name of any addressable bit in the Lower 128 or SFR space. An I/O line (the LSB of Port 1, in this case) is set or cleared depending on whether the flag bit is 1 or 0.

The Carry bit in the PSW is used as the single-bit Accumulator of the Boolean processor. Bit instructions that refer to the Carry bit as C assemble as Carry-specific instructions (CLR C, etc.). The Carry bit also has a direct address, since it resides in the PSW register, which is bit-addressable.

Note that the Boolean instruction set includes ANL and ORL operations, but not the XRL (Exclusive OR) operation. An XRL operation is simple to implement in software. Suppose, for example, it is required to form the Exclusive OR of two bits:

C = bit1 .XRL. bit2

The software to do that could be as follows:

```
      MOV     C,bit1
      JNB     bit2,OVER
      CPL     C
OVER: (continue)
```

First, bit1 is moved to the Carry. If bit2 = 0, then C now contains the correct result. That is, bit1 .XRL. bit2 = bit1 if bit2 = 0. On the other hand, if bit2 = 1, C now contains the complement of the correct result. It need only be inverted (CPL C) to complete the operation.

This code uses the JNB instruction, one of a series of bit-test instructions which execute a jump if the addressed bit is set (JC, JB, JBC) or if the addressed bit is not set (JNC, JNB). In the above case, bit2 is being tested, and if bit2 = 0, the CPL C instruction is jumped over.

JBC executes the jump if the addressed bit is set, and also clears the bit. Thus a flag can be tested and cleared in one operation. All the PSW bits are directly addressable, so the Parity bit, or the general purpose flags, for example, are also available to the bit-test instructions.

Relative Offset

The destination address for these jumps is specified to the assembler by a label or by an actual address in Program memory. However, the destination address assembles to a relative offset byte. This is a signed (two's complement) offset byte which is added to the PC in two's complement arithmetic if the jump is executed. The range of the jump is therefore –128 to +127 Program Memory bytes relative to the first byte following the instruction.

Table 5. 80C51 Lookup Table Read Instructions

MNEMONIC	OPERATION	EXECUTION TIME (μs)
MOVC A,@A+DPTR	Read program memory at (A + DPTR)	2
MOVC A,@A+PC	Read program memory at (A + PC)	2

Table 6. 80C51 Boolean Instructions

MNEMONIC		OPERATION	EXECUTION TIME (μs)
ANL	C,bit	C = C.AND.bit	2
ANL	C,/bit	C = C.AND..NOT.bit	2
ORL	C,bit	C = C.OR.bit	2
ORL	C,/bit	C = C.OR..NOT.bit	2
MOV	C,bit	C = bit	1
MOV	bit,C	bit = C	2
CLR	C	C = 0	1
CLR	bit	bit = 0	1
SETB	C	C = 1	1
SETB	bit	bit = 1	1
CPL	C	C = .NOT.C	1
CPL	bit	bit = .NOT.bit	1
JC	rel	Jump if C = 1	2
JNC	rel	Jump if C = 0	2
JB	bit,rel	Jump if bit = 1	2
JNB	bit,rel	Jump if bit = 0	2
JBC	bit,rel	Jump if bit = 1; CLR bit	2

Jump Instructions

Table 7 shows the list of unconditional jumps with execution time for a 12MHz clock.

The table lists a single "JMP addr" instruction, but in fact there are three SJMP, LJMP, and AJMP, which differ in the format of the destination address. JMP is a generic mnemonic which can be used if the programmer does not care which way the jump is encoded.

The SJMP instruction encodes the destination address as a relative offset, as described above. The instruction is 2 bytes long, consisting of the opcode and the relative offset byte. The jump distance is limited to a range of −128 to +127 bytes relative to the instruction following the SJMP.

The LJMP instruction encodes the destination address as a 16-bit constant. The instruction is 3 bytes long, consisting of the opcode and two address bytes. The destination address can be anywhere in the 64k Program Memory space.

The AJMP instruction encodes the destination address as an 11-bit constant. The instruction is 2 bytes long, consisting of the opcode, which itself contains 3 of the 11 address bits, followed by another byte containing the low 8 bits of the destination address. When the instruction is executed, these 11 bits are simply substituted for the low 11 bits in the PC. The high 5 bits stay the same. Hence the destination has to be within the same 2k block as the instruction following the AJMP.

In all cases the programmer specifies the destination address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the destination address into the correct format for the given instruction. If the format required by the instruction will not support the distance to the specified destination address, a "Destination out of range" message is written into the List file.

The JMP @A+DPTR instruction supports case jumps. The destination address is computed at execution time as the sum of the 16-bit DPTR register and the Accumulator. Typically, DPTR is set up with the address of a jump table. In a 5-way branch, for example, an integer 0 through 4 is loaded into the Accumulator. The code to be executed might be as follows:

```
MOV   DPTR,#JUMP TABLE
MOV   A,INDEX_NUMBER
RL    A
JMP   @A+DPTR
```

Table 7. Unconditional Jumps in 80C51 Devices

MNEMONIC		OPERATION	EXECUTION TIME (μs)
JMP	addr	Jump to addr	2
JMP	@A+DPTR	Jump to A + DPTR	2
CALL	addr	Call subroutine at addr	2
RET		Return from subroutine	2
RETI		Return from interrupt	2
NOP		No operation	1

The RL A instruction converts the index number (0 through 4) to an even number on the range 0 through 8, because each entry in the jump table is 2 bytes long:

```
JUMP TABLE:
        AJMP    CASE 0
        AJMP    CASE 1
        AJMP    CASE 2
        AJMP    CASE 3
        AJMP    CASE 4
```

Table 7 shows a single "CALL addr" instruction, but there are two of them, LCALL and ACALL, which differ in the format in which the subroutine address is given to the CPU. CALL is a generic mnemonic which can be used if the programmer does not care which way the address is encoded.

The LCALL instruction uses the 16-bit address format, and the subroutine can be anywhere in the 64k Program Memory space. The ACALL instruction uses the 11-bit format, and the subroutine must be in the same 2k block as the instruction following the ACALL.

In any case, the programmer specifies the subroutine address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the address into the correct format for the given instructions.

Subroutines should end with a RET instruction, which returns execution to the instruction following the CALL.

RETI is used to return from an interrupt service routine. The only difference between RET and RETI is that RETI tells the interrupt control system that the interrupt in progress is done. If there is no interrupt in progress at the time RETI is executed, then the RETI is functionally identical to RET.

Table 8 shows the list of conditional jumps available to the 80C51 user. All of these jumps specify the destination address by the relative offset method, and so are limited to a jump distance of –128 to +127 bytes from the instruction following the conditional jump instruction. Important to note, however, the user specifies to the assembler the actual destination address the same way as the other jumps: as a label or a 16-bit constant.

There is no Zero bit in the PSW. The JZ and JNZ instructions test the Accumulator data for that condition.

The DJNZ instruction (Decrement and Jump if Not Zero) is for loop control. To execute a loop N times, load a counter byte with N and terminate the loop with a DJNZ to the beginning of the loop, as shown below for N = 10.

```
        MOV             COUNTER,#10
LOOP: (begin loop)
        •
        •
        •
        (end loop)
        DJNZ            COUNTER,LOOP
        (continue)
```

The CJNE instruction (Compare and Jump if Not Equal) can also be used for loop control as in Figure 12. Two bytes are specified in the operand field of the instruction. The jump is executed only if the two bytes are not equal. In the example of Figure 12, the two bytes were data in R1 and the constant 2AH. The initial data in R1 was 2EH. Every time the loop was executed, R1 was decremented, and the looping was to continue until the R1 data reached 2AH.

Another application of this instruction is in "greater than, less than" comparisons. The two bytes in the operand field are taken as unsigned integers. If the first is less than the second, then the Carry bit is set (1). If the first is greater than or equal to the second, then the Carry bit is cleared.

CPU Timing

All 80C51 microcontrollers have an on-chip oscillator which can be used if desired as the clock source for the CPU. To use the on-chip oscillator, connect a crystal or ceramic resonator between the XTAL1 and XTAL2 pins of the microcontroller, and capacitors to ground as shown in Figure 13.

Examples of how to drive the clock with an external oscillator are shown in Figure 14. Note that in the NMOS devices (8051, etc.) the signal at the XTAL2 pin actually drives the internal clock generator. In the CMOS devices (80C51, etc.), the signal at the XTAL1 pin drives the internal clock generator. The internal clock generator defines the sequence of states that make up the 80C51 machine cycle.

Machine Cycles

A machine cycle consists of a sequence of 6 states, numbered S1 through S6. Each state time lasts for two oscillator periods. Thus a machine cycle takes 12 oscillator periods or 1μs if the oscillator frequency is 12MHz.

Each state is divided into a Phase 1 half and a Phase 2 half. Figure 15 shows that fetch/execute sequences in states and phases for various kinds of instructions. Normally two program fetches are generated during each machine cycle, even if the instruction being executed doesn't require it. If the instruction being executed doesn't need more code bytes, the CPU simply ignores the extra fetch, and the Program Counter is not incremented.

Execution of a one-cycle instruction (Figures 15a and 15b) begins during State 1 of the machine cycle, when the opcode is latched into the Instruction Register. A second fetch occurs during S4 of the same machine cycle. Execution is complete at the end of State 6 of this machine cycle.

The MOVX instructions take two machine cycles to execute. No program fetch is generated during the second cycle of a MOVX instruction. This is the only time program fetches are skipped. The fetch/execute sequence for MOVX instructions is shown in Figure 15d.

The fetch/execute sequences are the same whether the Program Memory is internal or external to the chip. Execution times do not depend on whether the Program Memory is internal or external.

Table 8. Conditional Jumps in 80C51 Devices

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION
		DIR	IND	REG	IMM	TIME (μs)
JZ rel	Jump if A = 0	Accumulator only				2
JNZ rel	Jump if A ≠ 0	Accumulator only				2
DJNZ <byte>,rel	Decrement and jump if not zero	X		X		2
CJNE A,<byte>,rel	Jump if A ≠ <byte>	X			X	2
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data		X	X		2

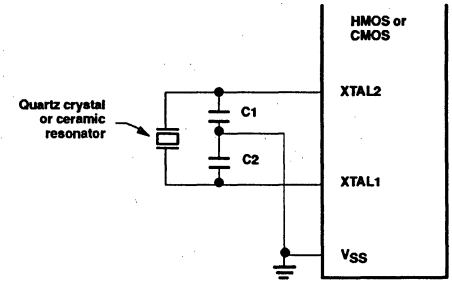
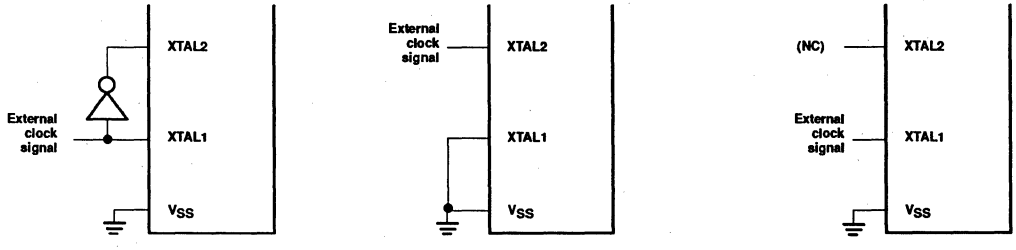


Figure 13. Using the On-Chip Oscillator



a. NMOS or CMOS

b. NMOS Only

c. CMOS Only

Figure 14. Using an External Clock

Figure 16 shows the signals and timing involved in program fetches when the Program Memory is external. If Program Memory is external, then the Program Memory read strobe $\overline{PSEN}$ is normally activated twice per machine cycle, as shown in Figure 16a. If an access to external Data Memory occurs, as shown in Figure 16b, two $\overline{PSEN}$s are skipped, because the address and data bus are being used for the Data Memory access.

Note that a Data Memory bus cycle takes twice as much time as a Program Memory bus cycle. Figure 16 shows the relative timing of the addresses being emitted at Ports 0 and 2, and of ALE and $\overline{PSEN}$. ALE is used to latch the low address byte from P0 into the address latch.

When the CPU is executing from internal Program Memory, $\overline{PSEN}$ is not activated, and program addresses are not emitted. However, ALE continues to be activated twice per machine cycle and so it is available as a clock output signal. Note, however, that one ALE is skipped during the execution of the MOVX instruction.

Interrupt Structure

The 80C51 and its ROMless and EPROM versions have 5 interrupt sources: 2 external interrupts, 2 timer interrupts, and the serial port interrupt.

What follows is an overview of the interrupt structure for the device. More detailed information for specific members of the 80C51 derivative family is provided in later chapters of this user's guide.

Interrupt Enables

Each interrupt source can be individually enabled or disabled by setting or clearing a bit in the SFR named IE (Interrupt Enable). This register also contains a global disable bit, which can be cleared to disable all interrupts at once. Figure 17 shows the IE register.

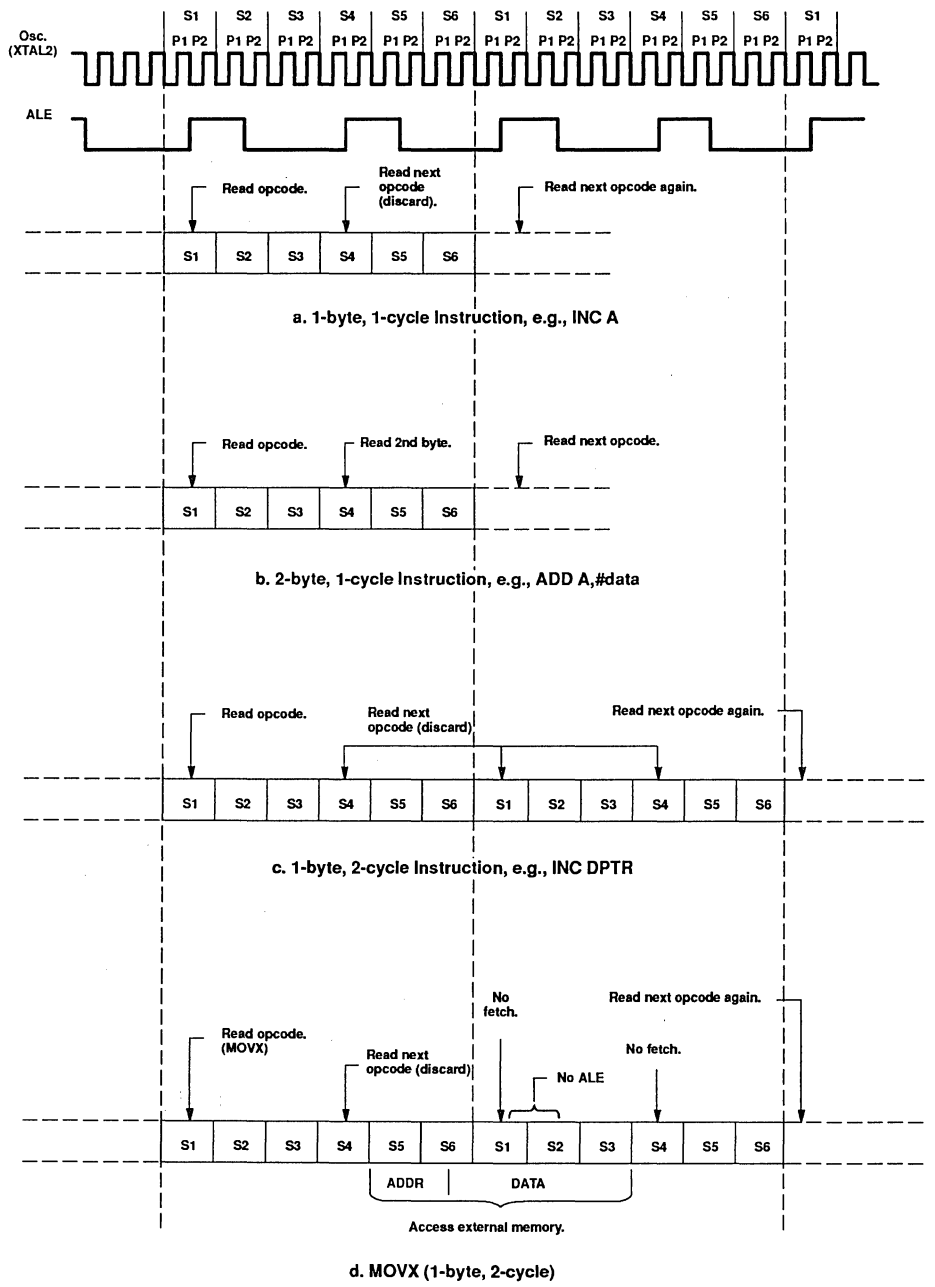
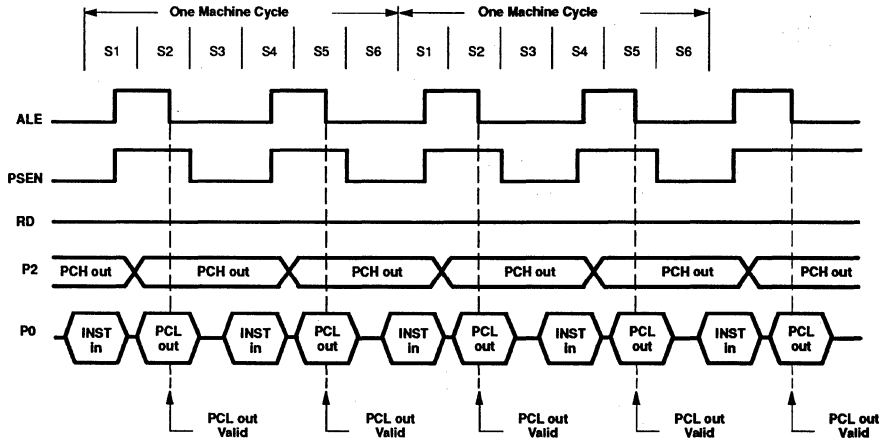
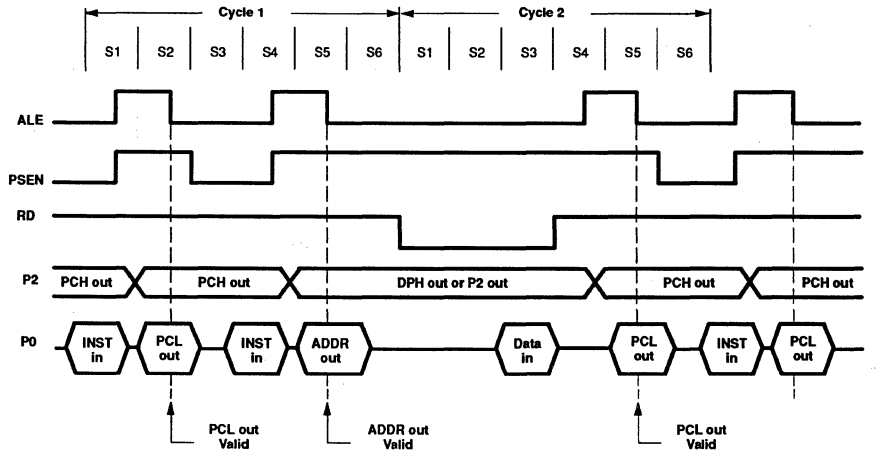


Figure 15. State Sequence in 80C51 Family Devices



a. Without a MOVX



b. With a MOVX

Figure 16. Bus Cycles in 80C51 Family Devices Executing from External Program Memory

(MSB)							(LSB)
$\overline{EA}$	X	X	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
$\overline{EA}$	IE.7	Disables all interrupts. If $\overline{EA}$ = 0, no interrupt will be acknowledged. If $\overline{EA}$ = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
	IE.6	Reserved.
	IE.5	Reserved.
ES	IE.4	Enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.
ET1	IE.3	Enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	Enables or disables External Interrupt 1. If EX1 = 0, External Interrupt 1 is disabled.
ET0	IE.1	Enables or disables the Timer 0 Overflow interrupt. If ET0 = 0, the Timer 0 interrupt is disabled.
EX0	IE.0	Enables or disables External Interrupt 0. If EX0 = 0, External Interrupt 0 is disabled.

Figure 17. Interrupt Enable (IE) Register

(MSB)							(LSB)
X	X	X	PS	PT1	PX1	PT0	PX0

Symbol	Position	Function
	IP.7	Reserved.
	IP.6	Reserved.
	IP.5	Reserved.
PS	IP.4	Defines the Serial Port interrupt priority level. PS = 1 programs it to the higher priority level.
PT1	IP.3	Defines theTimer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
PX1	IP.2	Defines the External Interrupt 1 priority level. PX1 = 1 programs it to the higher priority level.
PT0	IP.1	Enables or disables the Timer 0 Interrupt priority level. PT0 = 1 programs it to the higher priority level.
PX0	IP.0	Defines the External Interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

Figure 18. Interrupt Priority (IP) Register

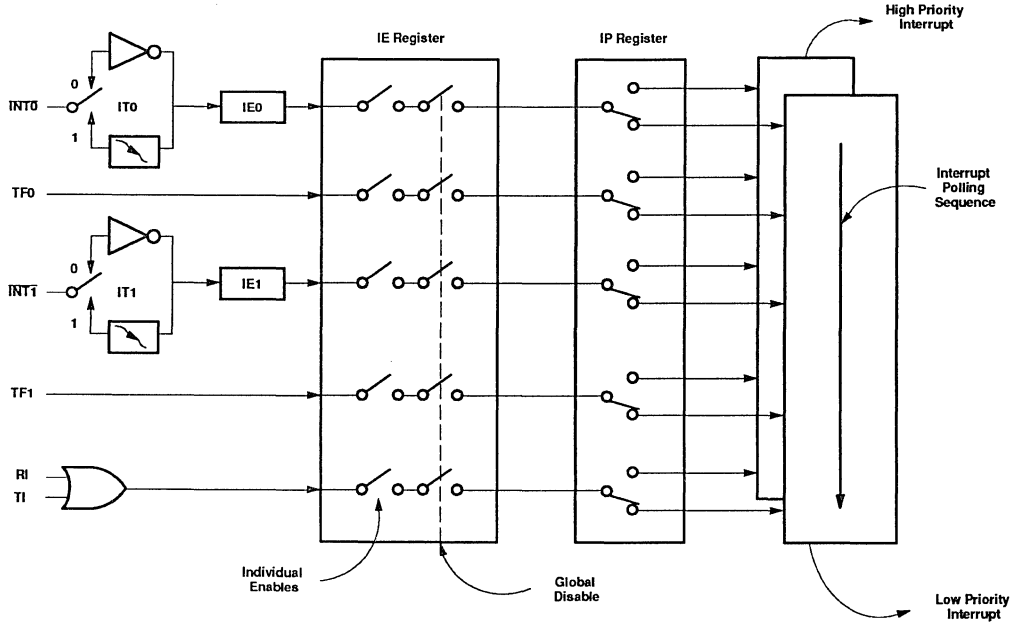


Figure 19. Interrupt Control System

Interrupt Priorities

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in the SFR named IP (Interrupt Priority). Figure 18 shows the IP register. A low-priority interrupt can be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

If two interrupt requests of different priority levels are received simultaneously, the request of higher priority is serviced. If interrupt requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence. Figure 19 shows how the IE and IP registers and the polling sequence work to determine which if any interrupt will be serviced.

In operation, all the interrupt flags are latched into the interrupt control system during State 5 of every machine cycle. The samples are polled during the following machine cycle. If the flag for an enabled interrupt is found to be set (1), the interrupt system generates an LCALL to the appropriate location in Program Memory, unless some other condition blocks the interrupt. Several conditions can block an interrupt, among them that an interrupt of equal or higher priority level is already in progress.

The hardware-generated LCALL causes the contents of the Program Counter to be pushed into the stack, and reloads the PC with the beginning address of the service routine. As previously noted (Figure 3), the service routine for each interrupt begins at a fixed location.

Only the Program Counter is automatically pushed onto the stack, not the PSW or any other register. Having only the PC automatically saved allows the programmer to decide how much time should be spent saving other registers. This enhances the interrupt response time, albeit at the expense of increasing the programmer's burden of responsibility. As a result, many interrupt functions that are typical in control applications toggling a port pin for example, or reloading a timer, or unloading a serial buffer can often be completed in less time than it takes other architectures to complete.

Simulating a Third Priority Level in Software

Some applications require more than two priority levels that are provided by on-chip hardware in 80C51 devices. In these cases, relatively simple software can be written to produce the same effect as a third priority level. First, interrupts that are to have higher priority than 1 are assigned to priority 1 in the Interrupt Priority (IP) register. The service routines for priority 1 interrupts that are supposed to be interruptable by priority 2 interrupts are written to include the following code:

```
        PUSH   IE
        MOV    IE,#MASK
        CALL   LABEL
        **********************
        (execute service routine)
        **********************
        POP    IE
        RET
LABEL:  RETI
```

As soon as any priority interrupt is acknowledged, the Interrupt Enable (IE) register is redefined so as to disable all but priority 2 interrupts. Then a CALL to LABEL executes the RETI instruction, which clears the priority 1 interrupt-in-progress flip-flop. At this point any priority 1 interrupt that is enabled can be serviced, but only priority 2 interrupts are enabled.

POPing IE restores the original enable byte. Then a normal RET (rather than another RETI) is used to terminate the service routine. The additional software adds 10µs (at 12MHz) to priority 1 interrupts.

80C51 family hardware description

HARDWARE DESCRIPTION

This chapter provides a detailed description of the 80C51 microcontroller (see Figure 1). Included in this description are:

- The port drivers and how they function both as ports and, for Ports 0 and 2, in bus operations
- The Timers/Counters
- The Serial Interface
- The Interrupt System
- Reset
- The Reduced Power Modes in CMOS devices
- The EPROM version of the 80C51

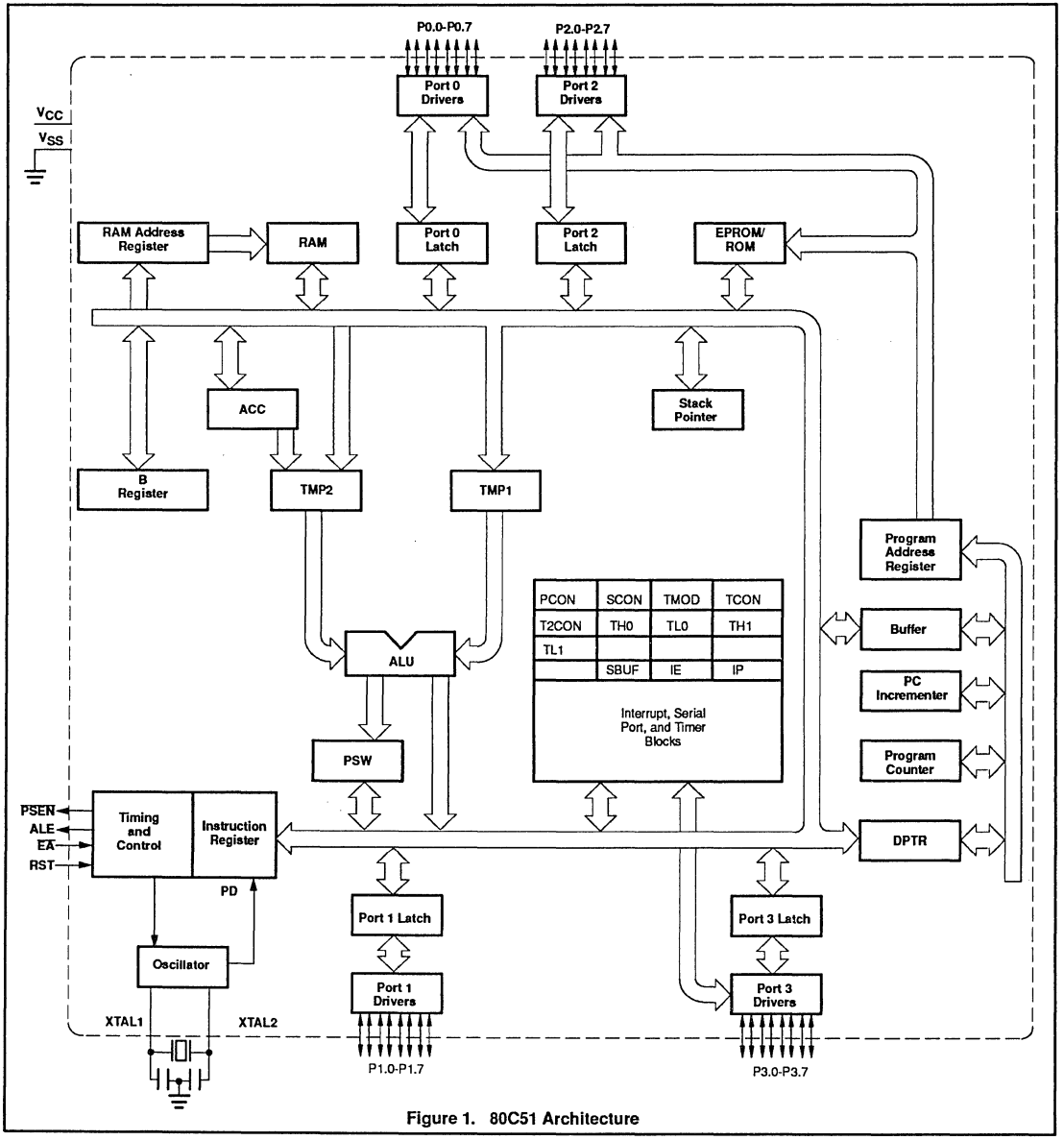


Figure 1. 80C51 Architecture

Special Function Registers

A Map of the on-chip memory area called the Special Function Register (SFR) space is shown in Figure 2.

Note that in the SFRs not all of the addresses are occupied. Unoccupied addresses are not implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have no effect.

User software should not write 1s to these unimplemented locations, since they may be used in other 80C51 Family derivative products to invoke new features. The functions of the SFRs are described in the text that follows.

Accumulator
ACC is the Accumulator register. The mnemonics for Accumulator-Specific instructions, however, refer to the Accumulator simply as A.

B Register
The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

Program StatusWord
The PSW register contains program status information as detailed in Figure 3.

Stack Pointer
The Stack Pointer register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at locations 08H.

Data Pointer
The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

Ports 0 to 3
P0, P1, P2, and P3 are the SFR latches of Ports 0, 1, 2, and 3, respectively. Writing a one to a bit of a port SFR (P0, P1, P2, or P3) causes the corresponding port output pin to switch high. Writing a zero causes the port output pin to switch low. When used as an input, the external state of a port pin will be held in the port SFR (i.e., if the external state of a pin is low, the corresponding port SFR bit will contain a 0; if it is high, the bit will contain a 1).

Serial Data Buffer
The Serial Buffer is actually two separate registers, a transmit buffer and a receive buffer. When data is moved to SBUF, it goes to the transmit buffer and is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

Timer Registers Basic to 80C51
Register pairs (TH0, TL0), and (TH1, TL1) are the 16-bit Counting registers for Timer/Counters 0 and 1, respectively.

Control Register for the 80C51
Special Function Registers IP, IE, TMOD, TCON, SCON, and PCON contain control and status bits for the interrupt system, the Timer/Counters, and the serial port. They are described in later sections.

Port Structures and Operation

All four ports in the 80C51 are bidirectional. Each consists of a latch (Special Function Registers P0 through P3), an output driver, and an input buffer.

The output drivers of Ports 0 and 2, and the input buffers of Port 0, are used in accesses to external memory. In this application, Port 0 outputs the low byte of the external memory address, time-multiplexed with the byte being written or read.

Port 2 outputs the high byte of the external memory address when the address is 16 bits wide. Otherwise, the Port 2 pins continue to emit the P2 SFR content.

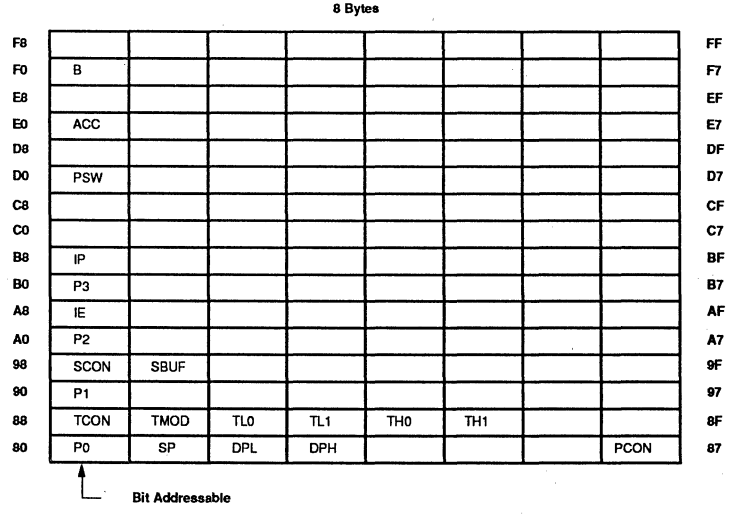
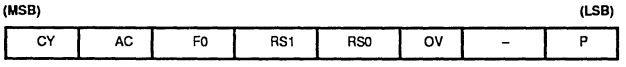


Figure 2. 80C51 SFR Memory Map



Symbol	Position	Name and Significance
CY	PSW.7	Carry flag.
AC	PSW.6	Auxiliary Carry flag. (For BCD operations.)
F0	PSW.5	Flag 0. (Available to the user for general purposes.)
RS1 RS0	PSW.4 PSW.3	Register bank select control bits 1 and 0. Set/cleared by software to determine working register bank (see Note).
OV	PSW.2	Overflow flag.
–	PSW.1	User-definable flag.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the Accumulator, i.e., even parity.

NOTE:
The contents of (RS1, RS0) enable the working register banks as follows:

(0,0)—Bank 0 (00H-07H)
(0,1)—Bank 1 (08H-0FH)
(1,0)—Bank 2 (10H-17H)
(1,1)—Bank 3 (18H-17H)

Figure 3. Program Status Word (PSW) Register

All the Port 3 pins are multifunctional. They are not only port pins, but also serve the functions of various special features as listed below:

Port Pin	Alternate Function
P3.0	RxD (serial input port)
P3.1	TxD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt)
P3.3	$\overline{INT1}$ (external interrupt)
P3.4	T0 (Timer/Counter 0 external input)
P3.5	T1 (Timer/Counter 1 external input)
P3.6	$\overline{WR}$ (external Data Memory write strobe)
P3.7	$\overline{RD}$ (external Data Memory read strobe)

The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise the port pin remains at 0.

I/O Configurations

Figure 4 shows a functional diagram of a typical bit latch and I/O buffer in each of the four ports. The bit latch (one bit in the port's SFR) is represented as a Type D flip-flop, which will clock in a value from the internal bus in response to a "write to latch" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to a "read pin" signal from the CPU. Some instructions that read a port activate the "read latch" signal, and others activate the "read pin" signal.

As shown in Figure 4, the output drivers of Port 0 and 2 are switchable to an internal ADDR and ADDR/DATA bus by an internal CONTROL signal for use in external memory accesses. During external memory accesses, the P2 SFR remains unchanged, but the P0 SFR gets 1s written to it.

Also shown in Figure 4 is that if a P3 bit latch contains a 1, then the output level is controlled by the signal labeled "alternate output function." The actual P3.X pin level is always available to the pin's alternate input function, if any.

Ports 1, 2, and 3 have internal pullups, and Port 0 has open drain outputs. Each I/O line can be independently used as an input or an output. (Port 0 and 2 may not be used as general purpose I/O when being used as the ADDR/DATA BUS for external memory during normal operation.) To be used as an input, the port bit latch must contain a 1, which turns off the output driver FET. Then, for Ports 1, 2, and 3, the pin is pulled high by a weak internal pullup, and can be pulled low by an external source.

Port 0 differs in that its internal pullups are not active during normal port operation. The pullup FET in the P0 output driver (see Figure 4) is used only when the port is emitting 1s during external memory accesses. Otherwise the pullup FET is off. Consequently P0 lines that are being used as output port lines are open drain. Writing a 1 to the bit latch leaves both output FETs off, so the pin floats. In that condition it can be used as a high-impedance input.

Because Ports 1, 2, and 3 have fixed internal pullups, they are sometimes called "quasi-bidirectional" ports. When configured as inputs they pull high and will source current (I_{IL}, in the data sheets) when externally pulled low. Port 0, on the other hand, is considered "true" bidirectional, because when configured as an input it floats.

All the port latches in the 80C51 have 1s written to them by the reset function. If a 0 is subsequently written to a port latch, it can be reconfigured as an input by writing a 1 to it.

Writing to a Port

In the execution of an instruction that changes the value in a port latch, the new value arrives at the latch during S6P2 of the final cycle of the instruction. However, port latches are in fact sampled by their output buffers only during Phase 1 of an clock period. (During Phase 2 the output buffer holds the value it saw during the previous Phase 1). Consequently, the new value in the port latch won't actually appear at the output pin until the next Phase 1, which will be at S1P1 of the next machine cycle.

If the change requires a 0-to-1 transition in Port 1, 2, or 3, an additional pullup is turned on during S1P1 and S1P2 of the cycle in which the transition occurs. This is done to increase the transition speed. The extra pullup can source about 100 times the current that the normal pullup can. It should be noted that the internal pullups are field-effect transistors, not linear resistors. The pullup arrangements are shown in Figure 5.

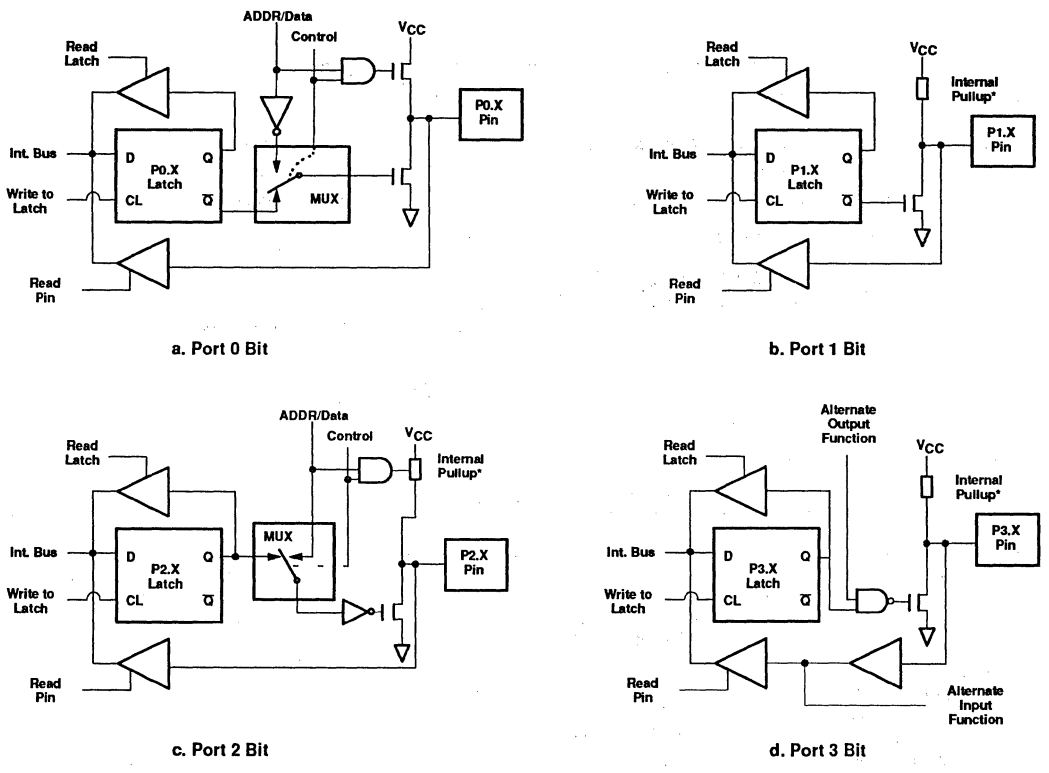


Figure 4. 80C51 Port Bit Latches and I/O Buffers

In the NMOS 8051 part, the fixed part of the pullup is a depletion mode transistor with the gate wired to the source. This transistor will allow the pin to source about 0.25mA when shorted to ground. In parallel with the fixed pullup is an enhancement mode transistor, which is activated during S1 whenever the port bit does a 0-to-1 transition. During this interval, if the port pin is shorted to ground, this extra transistor will allow the pin to source an additional 30mA.

In the CMOS 80C51, the pullup consists of three pFETs. It should be noted that an n-channel FET (nFET) is turned on when a logical 1 is applied to its gate, and is turned off when a logical 0 is applied to its gate. A p-channel FET (pFET) is the opposite: it is on when its gate sees a 0, and off when its gate sees a 1.

pFET1 in Figure 5 is the transistor that is turned on for 2 oscillator periods after a 0-to-1 transition in the port latch. While it's on, it turns on pFET3 (a weak pullup), through the inverter. This inverter and pPET form a latch which holds the 1.

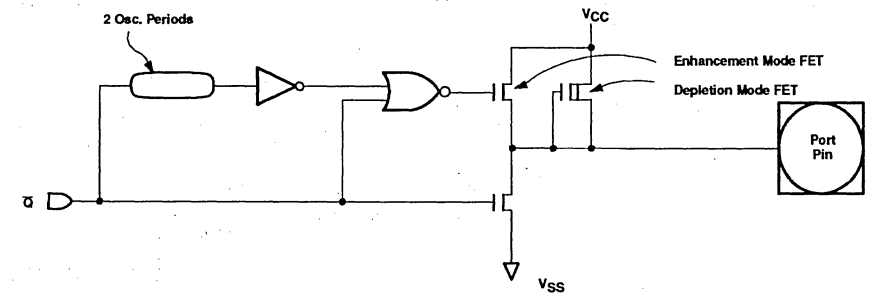
Note that if the pin is emitting a 1, a negative glitch on the pin from some external source can turn off pFET3, causing the pin to go into a float state. pFET2 is a very weak pullup which is on whenever the nFET is off, in traditional CMOS style. It's only about 1/10 the strength of pFET1. Its function is to restore a 1 to the pin in the event the pin had a 1 and lost it to a glitch.

Port Loading and Interfacing

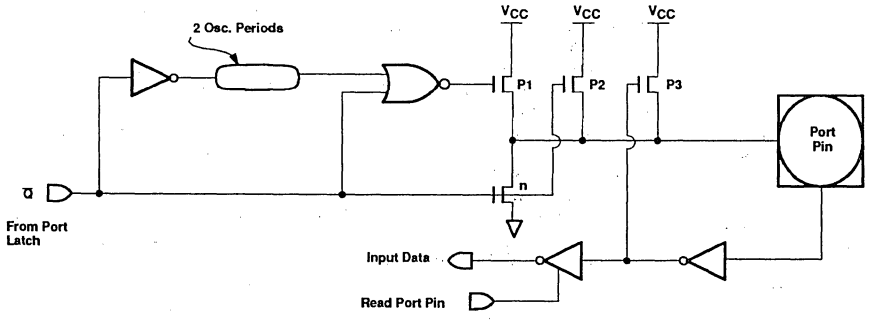
The output buffers of Ports 1, 2, and 3 can each drive 4 LS TTL inputs. These ports on NMOS versions can be driven in a normal manner by a TTL or NMOS circuit. Both NMOS and CMOS pins can be driven by open-collector and open-drain outputs, but note that 0-to-1 transitions will not be fast.

In the NMOS device, if the pin is driven by an open-collector output, a 0-to-1 transition will have to be driven by the relatively weak depletion mode FET in Figure 5a. In the CMOS device, an input 0 turns off pullup pFET3, leaving only the very weak pullup pFET2 to drive the transition.

Port 0 output buffers can each drive 8 LS TTL inputs. They do, however, require external pullups to drive NMOS inputs, except when being used as the ADDRESS/DATA bus for external memory.



a. NMOS Configuration.
The enhancement mode transistor is turned on for 2 oscillator periods after Q makes a 1-to-0 transition.



b. CMOS Configuration.
pFET1 is turned on for 2 oscillator periods after Q makes a 1-to-0 transition.
During this time, pFET1 also turns on pFET3 through the inverter to form a latch which holds the 1. pFET2 is also on.

Figure 5. Ports 1 and 3 NMOS and CMOS Internal Pullup Configurations
Port 2 is similar except that it holds the strong pullup on while emitting 1s that are address bits. (See *Accessing External Memory*.)

Read-Modify-Write Feature

Some instructions that read a port read the latch and others read the pin. Which ones do which? The instructions that read the latch rather than the pin are the ones that read a value, possibly change it, and then rewrite it to the latch. These are called "read-modify-write" instructions. The instructions listed below are read-modify-write instructions. When the destination operand is a port, or a port bit, these instructions read the latch rather than the pin:

ANL	(logical AND, e.g., ANL P1,A)
ORL	(logical OR, e.g., ORL P2,A)
XRL	(logical EX-OR, e.g., XRL P3,A)
JBC	(jump if bit = 1 and clear bit, e.g., JBC P1.1,LABEL)
CPL	(complement bit, e.g., CPL P3.0)
INC	(increment, e.g., INC P2)
DEC	(decrement, e.g., DEC P2)
DJNZ	(decrement and jump if not zero, e.g., DJNZ P3,LABEL)
MOV,PX,Y,C	(move carry bit to bit Y of Port X)
CLR PX.Y	(clear bit Y of Port X)
SET PX Y	(set bit Y of Port X)

It is not obvious that the last three instructions in this list are read-modify-write instructions, but they are. They read the port byte, all 8 bits, modify the addressed bit, then write the new byte back to the latch.

The reason that read-modify-write instructions are directed to the latch rather than the pin is to avoid a possible misinterpretation of the voltage level at the pin. For example, a port bit might be used to drive the base of a transistor. When a 1 is written to the bit, the transistor is turned on. If the CPU then reads the same port bit at the pin rather than the latch, it will read the base voltage of the transistor and interpret it as a 0. Reading the latch rather than the pin will return the correct value of 1.

Accessing External Memory

Accesses to external memory are of two types: accesses to external Program Memory and accesses to external Data Memory. Accesses to external Program Memory use signal $\overline{PSEN}$ (program store enable) as the read strobe. Accesses to external Data Memory use $\overline{RD}$ or $\overline{WR}$ (alternate functions of P3.7 and P3.6) to strobe the memory. Fetches from external Program Memory always use a 16-bit address. Accesses to external Data Memory can use either a 16-bit address (MOVX @ DPTR) or an 8-bit address (MOVX @Ri).

Whenever a 16-bit address is used, the high byte of the address comes out on Port 2, where it is held for the duration of the read or write cycle. Note that the Port 2 drivers use the strong pullups during the entire time that they are emitting address bits that are 1s. This is during the execution of a MOVX @DPTR instruction. During this time the Port 2 latch (the Special Function Register) does not have to contain 1s, and the contents of the Port 2 SFR are not modified. If the external memory cycle is not immediately followed by another external memory cycle, the undisturbed contents of the Port 2 SFR will reappear in the next cycle.

If an 8-bit address is being used (MOVX @Ri), the contents of the Port 2 SFR remain at the Port 2 pins throughout the external memory cycle. This will facilitate paging.

In any case, the low byte of the address is time-multiplexed with the data byte on Port 0. The ADDR/DATA signals drive both FETs in the Port 0 output buffers. Thus, in this application the Port 0 pins are not open-drain outputs, and do not require external pullups. ALE (Address Latch Enable) should be used to capture the address byte into an external latch. The address byte is valid at the negative transition of ALE. Then, in a write cycle, the data byte to be written appears on Port 0 just before $\overline{WR}$ is activated, and remains there until after $\overline{WR}$ is deactivated. In a read cycle, the incoming byte is accepted at Port 0 just before the read strobe is deactivated.

During any access to external memory, the CPU writes 0FFH to the Port 0 latch (the Special Function Register), thus obliterating whatever information the Port 0 SFR may have been holding.

External Program Memory is accessed under two conditions: Whenever signal $\overline{EA}$ is active; or whenever the program counter (PC) contains a number that is larger than 0FFFH (in the 80C51).

This require that the ROMless versions have $\overline{EA}$ wired low to enable the lower 4k program bytes to be fetched from external memory.

When the CPU is executing out of external Program Memory, all 8 bits of Port 2 are dedicated to an output function and may not be used for general purpose I/O. During external program fetches they output the high byte of the PC. During this time the Port 2 drivers use the strong pullups to emit PC bits that are 1s.

Timer/Counters

The 80C51 has two 16-bit Timer/Counter registers: Timer 0 and Timer 1. Both can be configured to operate either as timers or event counters (see Figure 6).

In the "Timer" function, the register is incremented every machine cycle. Thus, one can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

In the "Counter" function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0 or T1. In this function, the external input is sampled during S5P2 of every machine cycle.

When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (24 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full cycle. In addition to the "Timer" or "Counter" selection, Timer 0 and Timer 1 have four operating modes from which to select.

Timer 0 and Timer 1

The "Timer" or "Counter" function is selected by control bits C/T in the Special Function Register TMOD. These two Timer/Counters have four operating modes, which are selected by bit-pairs (M1, M0) in TMOD. Modes 0, 1, and 2 are the same for both Timers/Counters. Mode 3 is different. The four operating modes are described in the following text.

Mode 0

Putting either Timer into Mode 0 makes it look like an 8048 Timer, which is an 8-bit Counter with a divide-by-32 prescaler. Figure 7 shows the Mode 0 operation as it applies to Timer 1.

In this mode, the Timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag TF1. The counted input is enabled to the Timer when TR1 = 1 and either GATE = 0 or $\overline{INT1}$ = 1. (Setting GATE = 1 allows the Timer to be controlled by external input $\overline{INT1}$, to facilitate pulse width measurements). TR1 is a control bit in the Special Function Register TCON (Figure 8). GATE is in TMOD.

The 13-bit register consists of all 8 bits of TH1 and the lower 5 bits of TL1. The upper 3 bits of TL1 are indeterminate and should be ignored. Setting the run flag (TR1) does not clear the registers.

Mode 0 operation is the same for the Timer 0 as for Timer 1. Substitute TR0, TF0, and $\overline{INT0}$ for the corresponding Timer 1 signals in Figure 7. There are two different GATE bits, one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

Mode 1

Mode 1 is the same as Mode 0, except that the Timer register is being run with all 16 bits.

Mode 2

Mode 2 configures the Timer register as an 8-bit Counter (TL1) with automatic reload, as shown in Figure 9. Overflow from TL1 not only sets TF1, but also reloads TL1 with the contents of TH1, which is preset by software. The reload leaves TH1 unchanged.

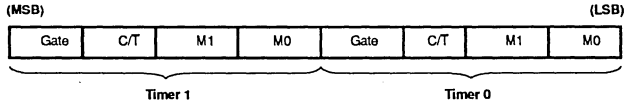
Mode 2 operation is the same for Timer/Counter 0.

Mode 3

Timer 1 in Mode 3 simply holds its count. The effect is the same as setting TR1 = 0.

Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. The logic for Mode 3 on Timer 0 is shown in Figure 10. TL0 uses the Timer 0 control bits: C/T, GATE, TR0, INT0, and TF0. TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from Timer 1. Thus, TH0 now controls the "Timer 1" interrupt.

Mode 3 is provided for applications requiring an extra 8-bit timer on the counter. With Timer 0 in Mode 3, an 80C51 can look like it has three Timer/Counters. When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3, or can still be used by the serial port as a baud rate generator, or in fact, in any application not requiring an interrupt.



Gate — Gating control when set. Timer/Counter "x" is enabled only while "INTx" pin is high and "TRx" control bit is set. When cleared Timer "x" is enabled whenever "TRx" control bit is set.

C/T — Timer or Counter Selector cleared for Timer operation (input from internal system clock). Set for Counter operation (input from "Tx" input pin).

M1	M0	Operating
0	0	8048 Timer "TLx" serves as 5-bit prescaler.
0	1	16-bit Timer/Counter "THx" and "TLx" are cascaded; there is no prescaler.
1	0	8-bit auto-reload Timer/Counter "THx" holds a value which is to be reloaded into "TLx" each time it overflows.
1	1	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits.
1	1	(Timer 1) Timer/Counter 1 stopped.

Figure 6. Timer/Counter Mode Control (TMOD) Register

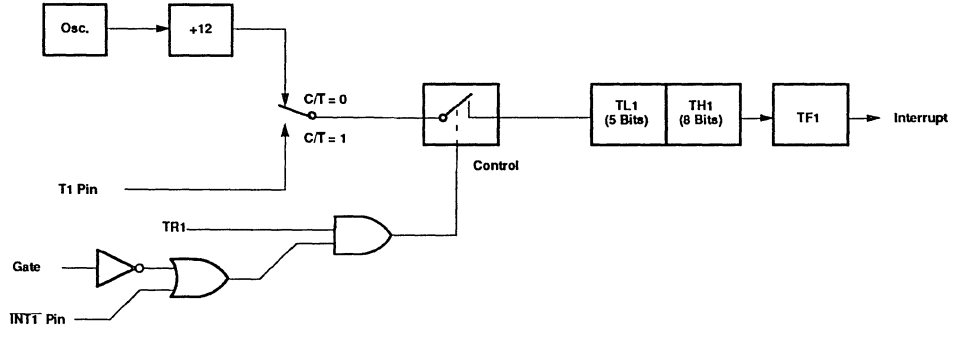
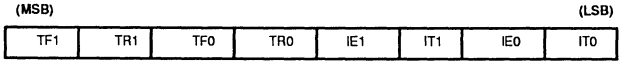


Figure 7. Timer/Counter 1 Mode 0: 13-Bit Counter



Symbol	Position	Name and Significance
TF1	TCON.7	Timer 1 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR1	TCON.6	Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
TF0	TCON.5	Timer 0 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR0	TCON.4	Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
IE1	TCON.3	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
IE0	TCON.1	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT0	TCON.0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.

Figure 8. Timer/Counter Control (TCON) Register

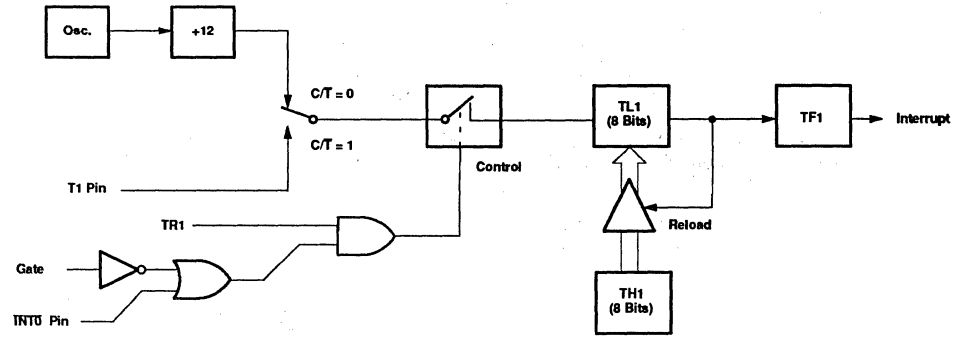


Figure 9. Timer/Counter 1 Mode 2: 8-Bit Auto-Load

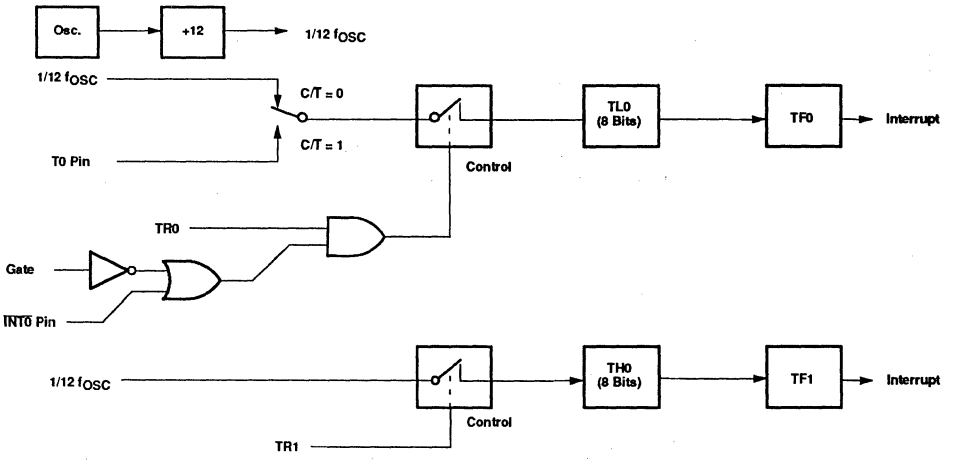


Figure 10. Timer/Counter 0 Mode 3: Two 8-Bit Counters

Standard Serial Interface

The serial port is full duplex, meaning it can transmit and receive simultaneously. It is also receive-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the register. (However, if the first byte still hasn't been read by the time reception of the second byte is complete, one of the bytes will be lost.) The serial port receive and transmit registers are both accessed at Special Function Register SBUF. Writing to SBUF loads the transmit register, and reading SBUF accesses a physically separate receive register.

The serial port can operate in 4 modes:

Mode 0: Serial data enters and exits through RxD. TxD outputs the shift clock. 8 bits are transmitted/received (LSB first). The baud rate is fixed at 1/12 the oscillator frequency.

Mode 1: 10 bits are transmitted (through TxD) or received (through RxD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in Special Function Register SCON. The baud rate is variable.

Mode 2: 11 bits are transmitted (through TxD) or received (through RxD): start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On Transmit, the 9th data bit (TB8 in SCON) can be assigned the value of 0 or 1. Or, for example, the parity bit (P, in the PSW) could be moved into TB8. On receive, the 9th data bit goes into RB8 in Special Function Register SCON, while the stop bit is ignored. The baud rate is programmable to either 1/32 or 1/64 the oscillator frequency.

Mode 3: 11 bits are transmitted (through TxD) or received (through RxD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). In fact, Mode 3 is the same as Mode 2 in all respects except baud rate. The baud rate in Mode 3 is variable.

In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in Mode 0 by the condition RI = 0 and REN = 1. Reception is initiated in the other modes by the incoming start bit if REN = 1.

Multiprocessor Communications

Modes 2 and 3 have a special provision for multiprocessor communications. In these modes, 9 data bits are received. The 9th one goes into RB8. Then comes a stop bit. The port can be programmed such that when the stop bit is received, the serial port interrupt will be activated only if RB8 = 1. This feature is enabled by setting bit SM2 in SCON. A way to use this feature in multiprocessor systems is as follows:

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM2 = 1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and see if it is being addressed. The addressed slave will clear its SM2 bit and prepare to receive the data bytes that will be coming. The slaves that weren't being addressed leave their SM2s set and go on about their business, ignoring the coming data bytes.

SM2 has no effect in Mode 0, and in Mode 1 can be used to check the validity of the stop bit. In a Mode 1 reception, if SM2 = 1, the receive interrupt will not be activated unless a valid stop bit is received.

Serial Port Control Register

The serial port control and status register is the Special Function Register SCON, shown in Figure 11. This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB8 and RB8), and the serial port interrupt bits (TI and RI).

Baud Rates

The baud rate in Mode 0 is fixed: Mode 0 Baud Rate = Oscillator Frequency / 12. The baud rate in Mode 2 depends on the value of bit SMOD in Special Function Register PCON. If SMOD = 0 (which is the value on reset), the baud rate is 1/64 the oscillator frequency. If SMOD = 1, the baud rate is 1/32 the oscillator frequency.

Mode 2 Baud Rate =

$$\frac{2^{SMOD}}{64} \times (\text{Oscillator Frequency})$$

In the 80C51, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate.

Using Timer 1 to Generate Baud Rates

When Timer 1 is used as the baud rate generator, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD as follows:

Mode 1, 3 Baud Rate =

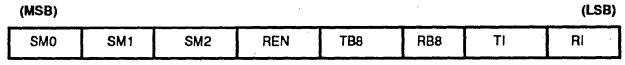
$$\frac{2^{SMOD}}{32} \times (\text{Timer 1 Overflow Rate})$$

The Timer 1 interrupt should be disabled in this application. The Timer itself can be configured for either "timer" or "counter" operation, and in any of its 3 running modes. In the most typical applications, it is configured for "timer" operation, in the auto-reload mode (high nibble of TMOD = 0010B). In that case the baud rate is given by the formula:

Mode 1, 3 Baud Rate =

$$\frac{2^{SMOD}}{32} \times \frac{\text{Oscillator Frequency}}{12 \times [256 - (TH1)]}$$

One can achieve very low baud rates with Timer 1 by leaving the Timer 1 interrupt enabled, and configuring the Timer to run as a 16-bit timer (high nibble of TMOD = 0001B), and using the Timer 1 interrupt to do a 16-bit software reload. Figure 12 lists various commonly used baud rates and how they can be obtained from Timer 1.



Where SM0, SM1 specify the serial port mode, as follows:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	shift register	$f_{OSC}/12$
0	1	1	8-bit UART	variable
1	0	2	9-bit UART	$f_{OSC}/64$ or $f_{OSC}/32$
1	1	3	9-bit UART	variable

- **SM2** enables the multiprocessor communication feature in modes 2 and 3. In mode 2 or 3, if SM2 is set to 1, then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1, then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0.
- **REN** enables serial reception. Set by software to enable reception. Clear by software to disable reception.
- **TB8** is the 9th data bit that will be transmitted in modes 2 and 3. Set or clear by software as desired.
- **RB8** in modes 2 and 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stopbit that was received in mode 0, RB8 is not used.
- **TI** is the transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software.
- **RI** is the receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software.

Figure 11. Serial Port Control (SCON) Register

Baud Rate	f_{OSC}	SMOD	Timer 1		
			C/T	Mode	Reload Value
Mode 0 Max: 1.67MHz	20MHz	X	X	X	X
Mode 2 Max: 625k	20MHz	1	X	X	X
Mode 1, 3 Max: 104.2k	20MHz	1	0	2	FFH
19.2k	11.059MHz	1	0	2	FDH
9.6k	11.059MHz	0	0	2	FDH
4.8k	11.059MHz	0	0	2	FAH
2.4k	11.059MHz	0	0	2	F4H
1.2k	11.059MHz	0	0	2	E8H
137.5	11.986M	0	0	2	1DH
110	6MHz	0	0	2	72H
110	12MHz	0	0	1	FEEBH

Figure 12. Timer 1 Generated Commonly Used Baud Rates

More About Mode 0

Serial data enters and exits through RxD. TxD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed a 1/12 the oscillator frequency.

Figure 13 shows a simplified functional diagram of the serial port in Mode 0, and associated timing.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal at S6P2 also loads a 1 into the 9th position of the transmit shift register and tells the TX Control block to commence a transmission. The internal timing is such that one full machine cycle will elapse between "write to SBUF" and activation of SEND.

SEND enables the output of the shift register to the alternate output function line of P3.0 and also enable SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK is low during S3, S4, and S5 of every machine cycle, and high during S6, S1, and S2. At S6P2 of every machine cycle in which SEND is active, the contents of the transmit shift are shifted to the right one position.

As data bits shift out to the right, zeros come in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position, is just to the left of the MSB, and all positions to the left of that contain zeros. This condition flags the TX Control block to do one last shift and then deactivate SEND and set T1. Both of these actions occur at S1P1 of the 10th machine cycle after "write to SBUF."

Reception is initiated by the condition REN = 1 and R1 = 0. At S6P2 of the next machine cycle, the RX Control unit writes the bits 11111110 to the receive shift register, and in the next clock phase activates RECEIVE.

RECEIVE enable SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK makes transitions at S3P1 and S6P1 of every machine cycle. At S6P2 of every machine cycle in which RECEIVE is active, the contents of the receive shift register are shifted to the left one position. The value that comes in from the right is the value that was sampled at the P3.0 pin at S5P2 of the same machine cycle.

As data bits come in from the right, 1s shift out to the left. When the 0 that was initially loaded into the rightmost position arrives at the leftmost position in the shift register, it flags the RX Control block to do one last shift and load SBUF. At S1P1 of the 10th machine cycle after the write to SCON that cleared RI, RECEIVE is cleared as RI is set.

More About Mode 1

Ten bits are transmitted (through TxD), or received (through RxD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in SCON. In the 80C51 the baud rate is determined by the Timer 1 overflow rate.

Figure 14 shows a simplified functional diagram of the serial port in Mode 1, and associated timings for transmit receive.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads a 1 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission actually commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal.)

The transmission begins with activation of SEND which puts the start bit at TxD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TxD. The first shift pulse occurs one bit time after that.

As data bits shift out to the right, zeros are clocked in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position is just to the left of the MSB, and all positions to the left of that contain zeros. This condition flags the TX Control unit to do one last shift and then deactivfate SEND and set TI. This occurs at the 10th divide-by-16 rollover after "write to SBUF."

Reception is initiated by a detected 1-to-0 transition at RxD. For this purpose RxD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written into the input shift register. Resetting the divide-by-16 counter aligns its rollovers with the boundaries of the incoming bit times.

The 16 states of the counter divide each bit time into 16ths. At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of RxD. The value accepted is the value that was seen in at least 2 of the 3 samples. This is done for noise rejection. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. This is to provide rejection of false start bits. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in mode 1 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated.:

1. R1 = 0, and
2. Either SM2 = 0,
   or the received stop bit = 1.

If either of these two conditions is not met, the received frame is irretrievably lost. If both conditions are met, the stop bit goes into RB8, the 8 data bits go into SBUF, and RI is activated. At this time, whether the above conditions are met or not, the unit goes back to looking for a 1-to-0 transition in RxD.

More About Modes 2 and 3

Eleven bits are transmitted (through TxD), or received (through RxD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On transmit, the 9th data bit (TB8) can be assigned the value of 0 or 1. On receive, the 9the data bit goes into RB8 in SCON. The baud rate is programmable to either 1/32 or 1/64 the oscillator frequency in Mode 2. Mode 3 may have a variable baud rate generated from Timer 1.

Figures 15 and 16 show a functional diagram of the serial port in Modes 2 and 3. The receive portion is exactly the same as in Mode 1. The transmit portion differs from Mode 1 only in the 9th bit of the transmit shift register.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads TB8 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal.)

The transmission begins with activation of SEND, which puts the start bit at TxD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TxD. The first shift pulse occurs one bit time after that. The first shift clocks a 1 (the stop bit) into the 9th bit position of the shift register. Thereafter, only zeros are clocked in. Thus, as data bits shift out to the right, zeros are clocked in from the left. When TB8 is at the output position of the shift register, then the stop bit is just to the left of TB8, and all positions to the left of that contain zeros. This condition flags the TX Control unit to do one last shift and then deactivate SEND and set TI. This occurs at the 11th divide-by-16 rollover after "write to SUBF."

Reception is initiated by a detected 1-to-0 transition at RxD. For this purpose RxD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written to the input shift register.

At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of R-D. The value accepted is the value that was seen in at least 2 of the 3 samples. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in Modes 2 and 3 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI.

The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated.

1. R1 = 0, and
2. Either SM2 = 0,
   or the received 9th data bit = 1.

If either of these conditions is not met, the received frame is irretrievably lost, and RI is not set. If both conditions are met, the received 9th data bit goes into RB8, and the first 8 data bits go into SBUF. One bit time later, whether the above conditions were met or not, the unit goes back to looking for a 1-to-0 transition at the RxD input.

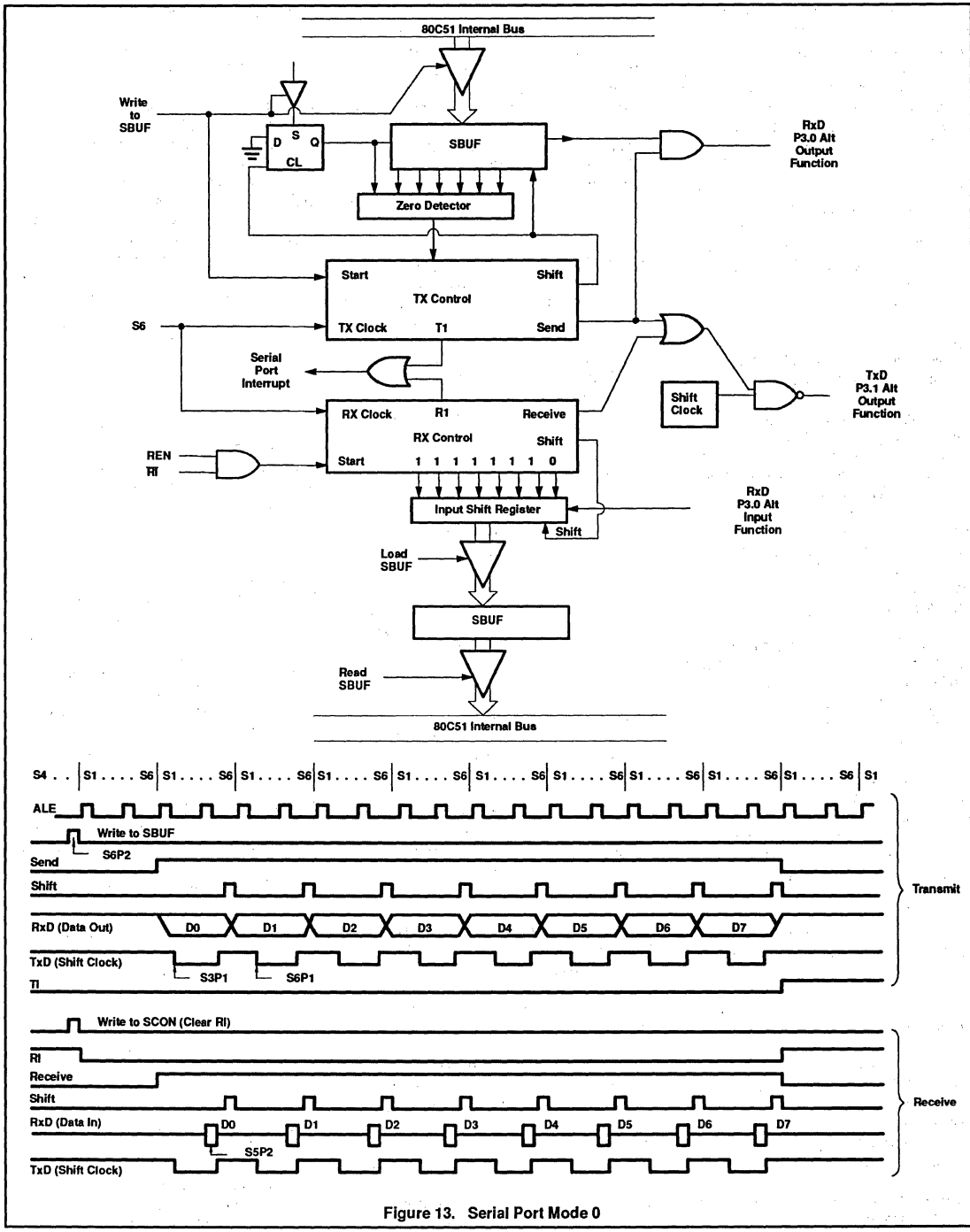


Figure 13. Serial Port Mode 0

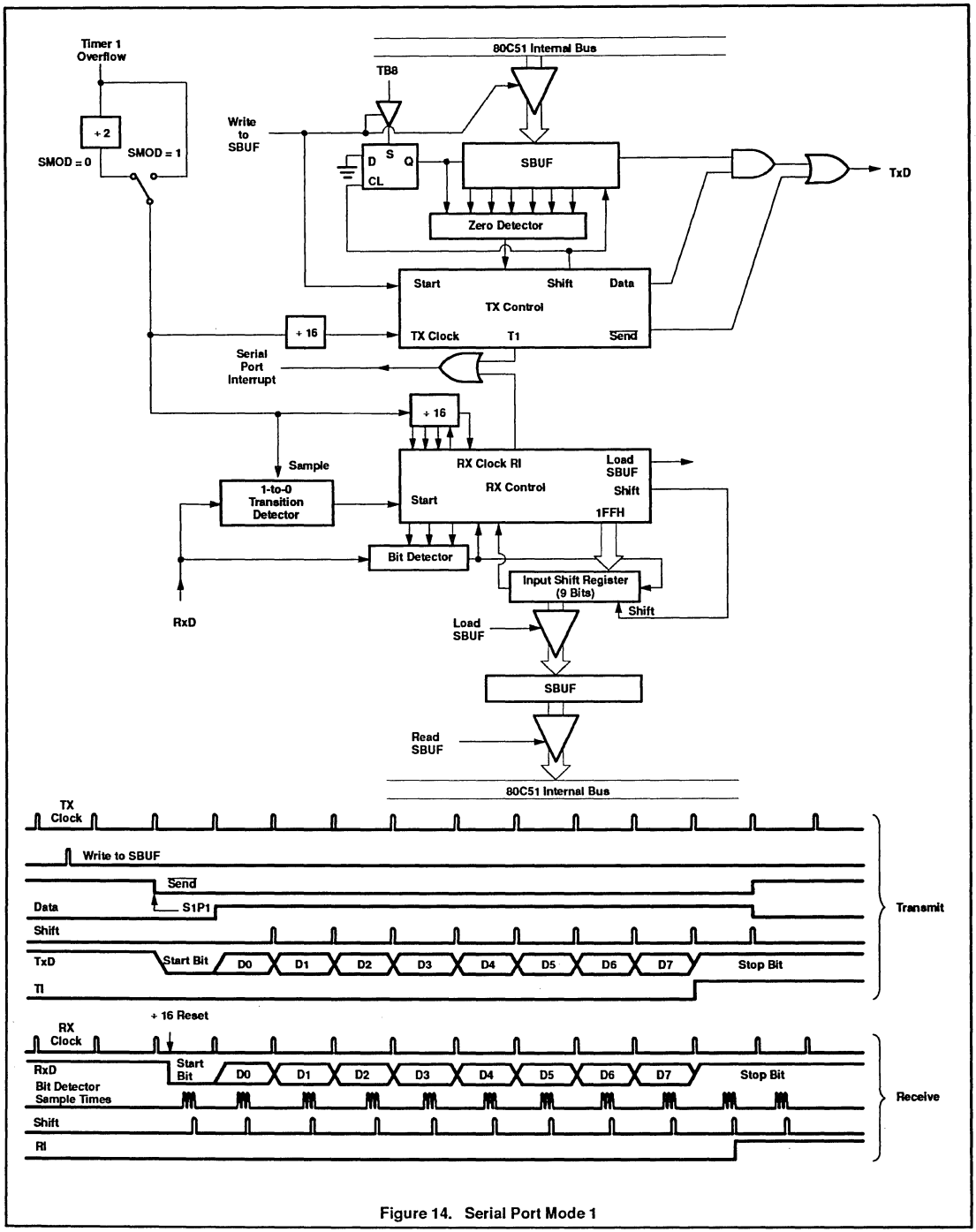


Figure 14. Serial Port Mode 1

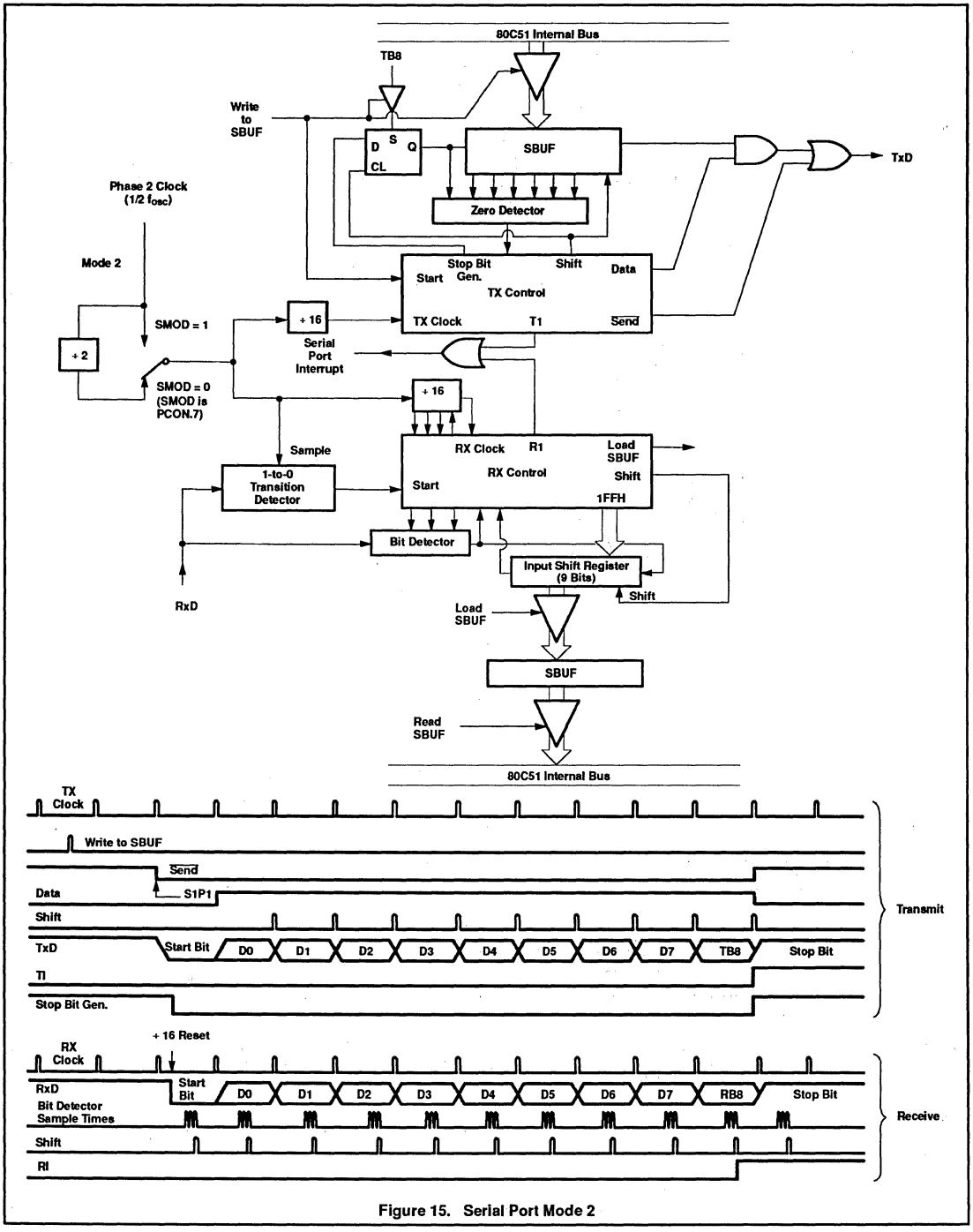


Figure 15. Serial Port Mode 2

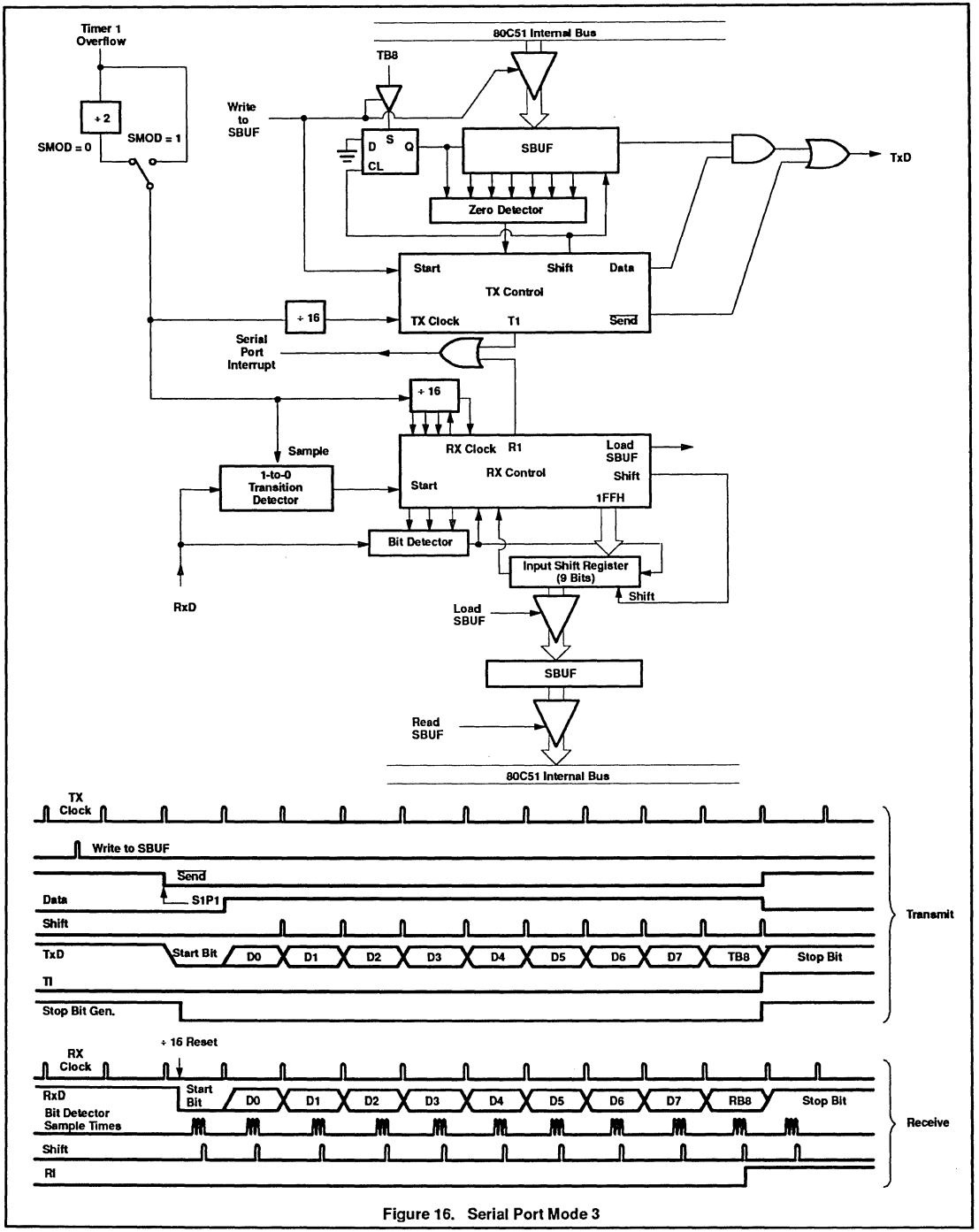


Figure 16. Serial Port Mode 3

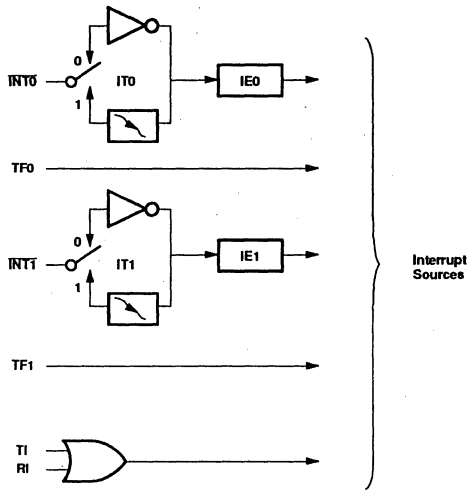


Figure 17. 80C51 Interrupt Sources

Interrupts

The 80C51 provides 5 interrupt sources. These are shown in Figure 17. The External Interrupts $\overline{INT0}$ and $\overline{INT1}$ can each be either level-activated or transition-activated, depending on bits IT0 and IT1 in Register TCON. The flags that actually generate these interrupts are bits IE0 and IE1 in TCON. When an external interrupt is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to only if the interrupt was transition-activated. If the interrupt was level-activated, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

The Timer 0 and Timer 1 Interrupts are generated by TF0 and TF1, which are set by a rollover in their respective Timer/Counter registers (except see Timer 0 in Mode 3). When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

The Serial Port Interrupt is generated by the logical OR of RI and TI. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine will normally have to determine whether it was RI or TI that generated the interrupt, and the bit will have to be cleared in software.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be canceled in software.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE (Figure 18). IE also contains a global disable bit, EA, which disables all interrupts at once.

Priority Level Structure

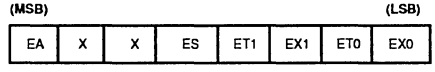
Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in Special Function Register IP (Figure 19). A low-priority interrupt can itself be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

If two request of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence as follows:

Source	Priority Within Level
1. IE0	(highest)
2. TF0	
3. IE1	
4. TF1	
5. RI+TI	(lowest)

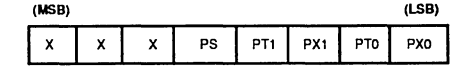
Note that the "priority within level" structure is only used to resolve simultaneous requests of the same priority level.

The IP register contains a number of unimplemented bits. IP.7, IP.6, and IP.5 are reserved in the 80C51. User software should not write 1s to these positions, since they may be used in other 8051 Family products.



Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	Reserved.
–	IE.5	Reserved.
ES	IE.4	Enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.
ET1	IE.3	Enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	Enables or disables External Interrupt.1. If EX1 = 0, External Interrupt 1 is disabled.
ET0	IE.1	Enables or disables the Timer 0 Overflow interrupt.. If ET0 = 0, the Timer 0 interrupt is disabled.
EX0	IE.0	Enables or disables External Interrupt. 0. If EX0 = 0, External Interrupt 0 is disabled.

Figure 18. Interrupt Enable Register (IE)



Symbol	Position	Function
–	IP.7	Reserved.
–	IP.6	Reserved.
–	IP.5	Reserved.
PS	IP.4	Defines the Serial Port interrupt priority level. PS = 1 programs it to the higher priority level.
PT1	IP.3	Defines theTimer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
PX1	IP.2	Defines the External Interrupt 1 priority level. PX1 =1 programs it to the higher priority level.
PT0	IP.1	Enables or disables the Timer 0 Interrupt priority level. PT0 = 1 programs it to the higher priority level.
PX0	IP.0	Defines the External Interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

Figure 19. Interrupt Priority Register (IP)

How Interrupts Are Handled

The interrupt flags are sampled at S5P2 of every machine cycle. The samples are polled during the following machine cycle. If one of the flags was in a set condition at S5P2 of the preceding cycle, the polling cycle will find it and the interrupt system will generate an LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1. An interrupt of equal or higher priority level is already in progress.
2. The current (polling) cycle is not the final cycle in the execution of the instruction in progress.
3. The instruction in progress is RETI or any write to the IE or IP registers.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress will be completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any access to IE or IP, then at least one more instruction will be executed before any interrupt is vectored to.

The polling cycle is repeated with each machine cycle, and the values polled are the values that were present at S5P2 of the previous machine cycle. Note that if an interrupt flag is active but not being responded to for one of the above conditions, if the flag is not still active when the blocking condition is removed, the denied interrupt will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

The polling cycle/LCALL sequence is illustrated in Figure 20.

Note that if an interrupt of higher priority level goes active prior to S5P2 of the machine cycle labeled C3 in Figure 20, then in accordance with the above rules it will be vectored to during C5 and C6, without any instruction of the lower priority routine having been executed.

Thus the processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate servicing routine. In some cases it also clears the flag that generated the interrupt, and in other cases it doesn't. It never clears the Serial Port flag. This has to be done in the user's software. It clears an external interrupt flag (IE0 or IE1) only if it was transition-activated. The hardware-generated LCALL pushes the contents of the Program Counter on to the stack (but it does not save the PSW) and reloads the PC with an address that depends on the source of the interrupt being vectored to, as shown below:

Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI+TI	0023H

Execution proceeds from that location until the RETI instruction is encountered. The RETI instruction informs the processor that this interrupt routine is no longer in progress, then pops the top two bytes from the stack and reloads the Program Counter. Execution of the interrupted program continues from where it left off.

Note that a simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking an interrupt was still in progress, making future interrupts impossible.

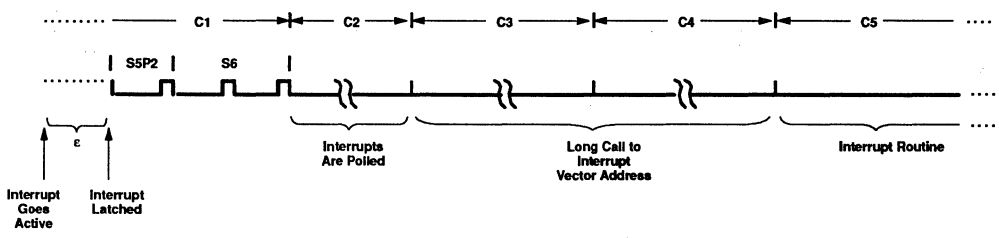


Figure 20. Interrupt Response Timing Diagram

External Interrupts

The external sources can be programmed to be level-activated or transition-activated by setting or clearing bit IT1 or IT0 in Register TCON. If ITx = 0, external interrupt x is triggered by a detected low at the INTx pin. If ITx = 1, external interrupt x is edge triggered. In this mode if successive samples of the INTx pin show a high in one cycle and a low in the next cycle, interrupt request flag IEx in TCON is set. Flag bit IEx then requests the interrupt.

Since the external interrupt pins are sampled once each machine cycle, an input high or low should hold for at least 12 oscillator periods to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin high for at least one cycle, and then hold it low for at least one cycle. This is done to ensure that the transition is seen so that interrupt request flag IEx will be set. IEx will be automatically cleared by the CPU when the service routine is called.

If the external interrupt is level-activated, the external source has to hold the request active until the requested interrupt is actually generated. Then it has to deactivate the request before the interrupt service routine is completed, or else another interrupt will be generated.

Response Time

The $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ levels are inverted and latched into IE0 and IE1 at S5P2 of every machine cycle. The values are not actually polled by the circuitry until the next machine cycle. If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two cycles. Thus, a minimum of three complete machine cycles elapse between activation of an external interrupt request and the beginning of execution of the first instruction of the service routine. Figure 20 shows interrupt response timings.

A longer response time would result if the request is blocked by one of the 3 previously listed conditions. If an interrupt of equal or higher priority level is already in progress, the additional wait time obviously depends on the nature of the other interrupt's service routine. If the instruction in progress is not in its final cycle, the additional wait time cannot be more the 3 cycles, since the longest instructions (MUL and DIV) are only 4 cycles long, and if the instruction in progress is RETI or an access to IE or IP, the additional wait time cannot be more than 5 cycles (a maximum of one more cycle to complete the instruction in progress, plus 4 cycles to complete the next instruction if the instruction is MUL or DIV).

Thus, in a single-interrupt system, the response time is always more than 3 cycles and less than 9 cycles.

Single-Step Operation

The 80C51 interrupt structure allows single-step execution with very little software overhead. As previously noted, an interrupt request will not be responded to while an interrupt of equal priority level is still in progress, nor will it be responded to after RETI until at least one other instruction has been executed. Thus, once an interrupt routine has been entered, it cannot be re-entered until at least one instruction of the interrupted program is executed. One way to use this feature for single-step operation is to program one of the external interrupts (e.g., INT0) to be level-activated. The service routine for the interrupt will terminate with the following code:

```
JNB   P3.2,$  ;Wait Till INT0 Goes High
JB    P3.2,$  ;Wait Till INT0 Goes Low
RETI          ;Go Back and Execute One
              Instruction
```

Now if the INT0 pin, which is also the P3.2 pin, is held normally low, the CPU will go right into the External Interrupt 0 routine and stay there until INT0 is pulsed (from low to high to low). Then it will execute RETI, go back to the task program, execute one instruction, and immediately re-enter the External Interrupt 0 routine to await the next pulsing of P3.2. One step of the task program is executed each time P3.2 is pulsed.

Reset

The reset input is the RST pin, which is the input to a Schmitt Trigger. A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. The CPU responds by generating an internal reset, with the timing shown in Figure 21.

The external reset signal is asynchronous to the internal clock. The RST pin is sampled during State 5 Phase 2 of every machine cycle. The port pins will maintain their current activities for 19 oscillator periods after a logic 1 has been sampled at the RST pin; that is, for 19 to 31 oscillator periods after the external reset signal has been applied to the RST pin.

The internal reset algorithm writes 0s to all the SFRs except the port latches, the Stack Pointer, and SBUF. The port latches are initialized to FFH, the Stack Pointer to 07H, and SBUF is indeterminate. Table 1 lists the SFR reset values. The internal RAM is not affected by reset. On power up the RAM content is indeterminate.

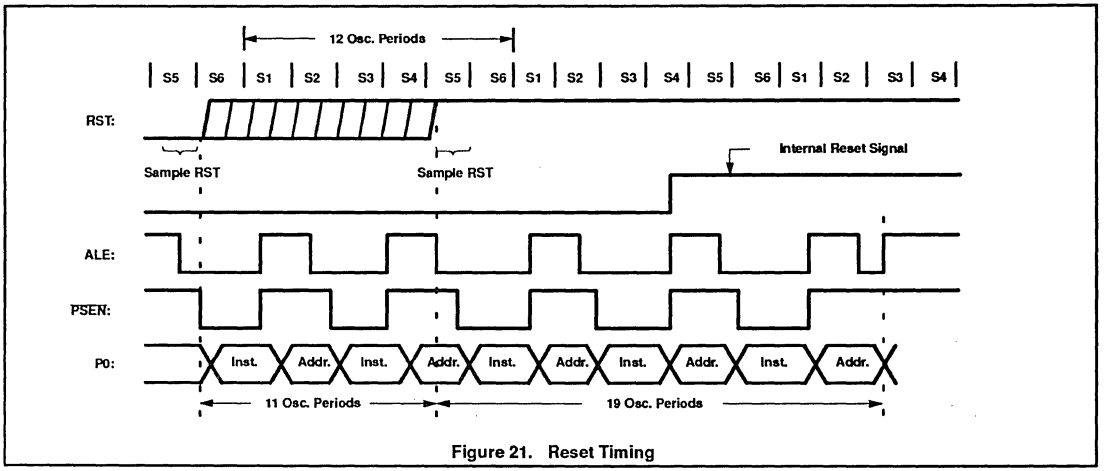


Figure 21. Reset Timing

Table 1. 80C51 SFR Reset Values

REGISTER	RESET VALUE
PC	000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0–P3	FFH
IP	XXX00000B
IE	0XX00000B
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
SCON	00H
SBUF	Indeterminate
PCON (NMOS)	0XXXXXXXB
PCON (CMOS)	0XXX0000B

Power-on Reset

An automatic reset can be obtained when V_{CC} is turned on by connecting the RST pin to V_{CC} through a 10µf capacitor and to V_{SS} through an 8.2k resistor, providing the V_{CC} rise time does not exceed 1 millisecond and the oscillator start-up time does not exceed 10 milliseconds. This power-on reset circuit is shown in Figure 22. The CMOS devices do not require the 8.2k pulldown resistor, although its presence does no harm.

When power is turned on, the circuit holds the RST pin high for an amount of time that depends on the value of the capacitor and the rate at which it charges. To ensure a good reset, the RST pin must be high long enough to allow the oscillator time to start-up (normally a few ms) plus two machine cycles.

Note that the port pins will be in a random state until the oscillator has started and the internal reset algorithm has written 1s to them.

With this circuit, reducing V_{CC} quickly to 0 causes the RST pin voltage to momentarily fall below 0V. However, this voltage is internally limited, and will not harm the device.

Power-Saving Modes of Operation

For applications where power consumption is critical the CMOS version provides power reduced modes of operation as a standard feature. The power down mode in NMOS devices is no longer a standard feature.

CMOS Power Reduction Mode

CMOS versions have two power reducing modes, Idle and Power Down. The input through which backup power is supplied during these operations is V_{CC}. Figure 23 shows the internal circuitry which implements these features. In the Idle modes (IDL = 1), the oscillator continues to run and the Interrupt, Serial Port, and Timer blocks continue to be clocked, but the clock signal is gated off to the CPU. In Power Down (PD = 1), the oscillator is frozen. The Idle and Power Down Modes are activated by setting bits in Special Function Register PCON. The address of this register is 87H. Figure 24 details its contents.

In the NMOS devices the PCON register only contains SMOD. The other four bits are implemented only in the CMOS devices. User software should never write 1s to unimplemented bits, since they may be used in other 80C51 Family products.

Idle Mode

An instruction that sets PCON.0 causes that to be the last instruction executed before going into the Idle mode, the internal clock signal is gated off to the CPU but not to the Interrupt, Timer, and Serial Port functions. The CPU status is preserved in its entirety; the Stack Pointer, Program Counter, Program Status Word, Accumulator, and all other registers maintain their data during Idle. The port pins hold the logical states they had at the time Idle was activated. ALE and $\overline{PSEN}$ hold at logic high levels.

There are two ways to terminate the Idle. Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, terminating the Idle mode. The interrupt will be serviced, and following RETI, the next instruction to be executed will be the one following the instruction that put the device into Idle.

The flag bits GF0 and GF1 can be used to give an indication if an interrupt occurred during normal operation or during an Idle. For example, an instruction that activates Idle can also set one or both flag bits. When Idle is terminated by an interrupt, the interrupt service routine can examine the flag bits. The other way of terminating the Idle mode is with a hardware reset. Since the clock oscillator is still running, the hardware reset needs to be held active for only two machine cycles (24 oscillator periods) to complete the reset.

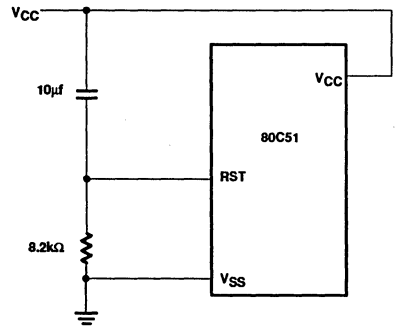


Figure 22. Power-On Reset Circuit

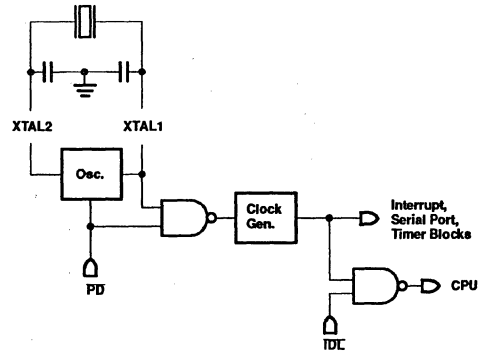


Figure 23. Idle and Power Down Hardware

(MSB) (LSB)

SMOD	–	–	–	GF1	GF0	PD	IDL

Symbol	Position	Name and Function
SMOD	PCON.7	Double Baud rate bit. When set to a 1 and Timer 1 is used to generate baud rate, and the Serial Port is used in modes 1, 2, or 3.
–	PCON.6	Reserved.
–	PCON.5	Reserved.
–	PCON.4	Reserved.
GF1	PCON.3	General-purpose flag bit.
GF0	PCON.2	General-purpose flag bit.
PD	PCON.1	Power-Down bit. Setting this bit activates power-down operation.
IDL	PCON.0	Idle mode bit. Setting this bit activates idle mode operation.

If 1s are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (0XXX0000). In the NMOS devices, the PCON register only contains SMOD. The other four bits are implemented only in the CMOS devices. User software should never write 1s to unimplemented bits, since they may be used in future products.

Figure 24. Power Control (PCON) Register

The signal at the RST pin clears the IDL bit directly and asynchronously. At this time the CPU resumes program execution from where it left off; that is, at the instruction following the one that invoked the Idle Mode. As shown in Figure 21, two or three machine cycles of program execution may take place before the internal reset algorithm takes control. On-chip hardware inhibits access to the internal RAM during this time, but access to the port pins is not inhibited. To eliminate the possibility of unexpected outputs at the port pins, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external Data RAM.

Power-Down Mode

An instruction that sets PCON.1 causes that to be the last instruction executed before going into the Power Down mode. In the Power Down mode, the on-chip oscillator is stopped. With the clock frozen, all functions are stopped, the contents of the on-chip RAM and Special Function Registers are maintained. The port pins output the values held by their respective SFRs. The ALE and $\overline{PSEN}$ output are held low.

The only exit from Power Down is a hardware reset. Reset redefines all the SFRs, but does not change the on-chip RAM.

In the Power Down mode of operation, V_{CC} can be reduced to as low as 2V. Care must be taken, however, to ensure that V_{CC} is not reduced before the Power Down mode is invoked, and that V_{CC} is restored to its normal operating level, before the Power Down mode is terminated. The reset that terminates Power Down also frees the oscillator. The reset should not be activated before V_{CC} is restored to its normal operating level, and must be held active long enough to allow the oscillator to restart and stabilize (normally less than 10ms).

ONCE Mode

The ONCE ("on-circuit emulation") mode facilitates testing and debugging of systems using the device without the device having to be removed from the circuit. The ONCE mode is invoked by:

1. Pull ALE low while the device in in reset and $\overline{PSEN}$ is high;
2. Hold ALE low as RST is deactivated.

While the device is in the ONCE mode, the Port 0 pins go into a float state, and the other port pins and ALE and $\overline{PSEN}$ are weakly pulled high. The oscillator circuit remains active. While the device is in this mode, an emulator or test CPU can be used to drive the circuit. Normal operation is restored after a normal reset is applied.

The On-Chip Oscillators

NMOS Version

The on-chip oscillator circuitry for the NMOSmembers of the 80C51 family is a single stage linear inverter (Figure 25), intended for use as a crystal-controlled, positive reactance oscillator (Figure 26). In this application the crystal is operated in its fundamental response mode as an inductive reactance in parallel resonance with capacitance external to the crystal.

The crystal specifications and capacitance values (C1 and C2 in Figure 26) are not critical. 30pF can be used in these positions at any frequency with good quality crystals. A ceramic resonator can be used in place of the crystal in cost-sensitive applications. When a ceramic resonator is used, C1 and C2 are normally selected to be of somewhat higher values, typically, 47pF. The manufacturer of the ceramic resonator should be consulted for recommendation on the values of these capacitors.

To drive the NMOS parts with an external clock source, apply the external clock signal to XTAL2, and ground XTAL1, as shown in Figure 27. A pullup resistor may be used (to increase noise margin), but is optional if V_{OH} of the driving gate exceeds the V_{IH} minimum specification of XTAL.

CMOS Versions

The on-chip oscillator circuitry for the 80C51, shown in Figure 28, consists of a single stage linear inverter intended for use as a crystal-controlled, positive reactance oscillator in the same manner as the NMOS parts. However, there are some important differences.

One difference is that the 80C51 is able to turn off its oscillator under software control (by writing a 1 to the PD bit in PCON). Another difference is that, in the 80C51, the internal clocking circuitry is driven by the signal at XTAL1, whereas in the NMOS versions it is by the signal at XTAL2.

The feedback resistor R_f in Figure 28 consists of paralleled n- and p-channel FETs controlled by the PD bit, such that R_f is opened when PD = 1. The diodes D1 and D2, which act as clamps to V_{CC} and V_{SS}, are parasitic to the R_f FETs. The oscillator can be used with the same external components as the NMOS versions, as shown in Figure 29. Typically, C1 = C2 = 30pF when the feedback element is a quartz crystal, and C1 = C2 = 47pF when a ceramic resonator is used.

To drive the CMOS parts with an external clock source, apply the external clock signal to XTAL1, and leave XTAL2 float, as shown in Figure 30.

The reason for this change from the way the NMOS part is driven can be seen by comparing Figures 26 and 28. In the NMOS devices the internal timing circuits are driven by the signal at XTAL2. In the CMOS devices the internal timing circuits are driven by the signal at XTAL1.

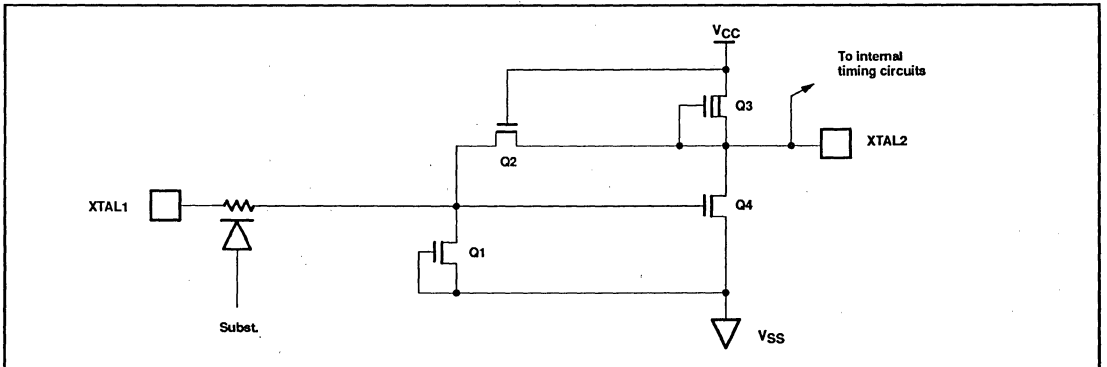


Figure 25. On-Chip Oscillator in the NMOS Version of the 8051 Family

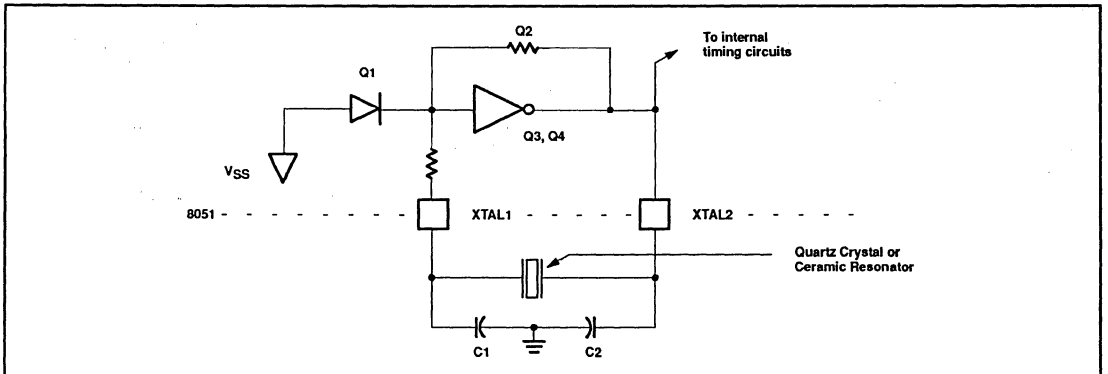


Figure 26. Using the NMOS On-Chip Oscillator

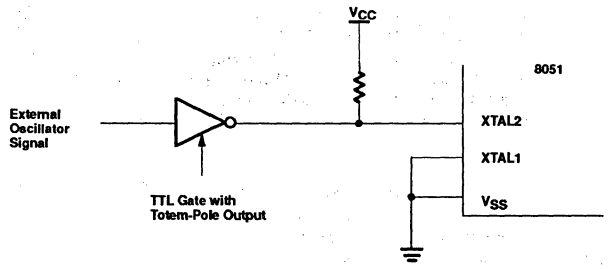


Figure 27. Driving the NMOS 8051 Family Parts with an External Clock

Internal Timing

Figures 31 through 34 show when the various strobe and port signals are clocked internally. The figures do not show rise and fall times of the signals, nor do they show propagation delays between the XTAL2 signal and events at other pins.

Rise and fall times are dependent on the external loading that each pin must drive. They are often taken to be something in the neighborhood of 10ns, measured between 0.8V and 2.0V.

Propagation delays are different for different pins. For a given pin they vary with pin loading, temperature, V_{CC}, and manufacturing lot. If the XTAL2 waveform is taken as the timing reference, prop delays may vary up to ±200%.

The AC Timings section of the data sheets do not reference any timing to the XTAL2 waveform. Rather, they relate the critical edges of control and input signals to each other. The timings published in the data sheets include the effects of propagation delays under the specified test conditions.

80C51 Pin Descriptions

ALE/PROG: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. ALE is emitted at a constant rate of 1/6 of the oscillator frequency, for external timing or clocking purposes, even when there are no accesses to external memory. (However, one ALE pulse is skipped during each access to external Data Memory.) This pin is also the program pulse input (PROG) during EPROM programming.

$\overline{PSEN}$: Program Store Enable is the read strobe to external Program Memory. When the device is executing out of external Program Memory, $\overline{PSEN}$ is activated twice each machine cycle (except that two $\overline{PSEN}$ activations are skipped during accesses to external Data Memory). $\overline{PSEN}$ is not activated when the device is executing out of internal Program Memory.

$\overline{EA}$/V_{PP}: When $\overline{EA}$ is held high the CPU executes out of internal Program Memory (unless the Program Counter exceeds 0FFFH in the 80C51). Holding $\overline{EA}$ low forces the CPU to execute out of external memory regardless of the Program Counter value. In the 80C31, $\overline{EA}$ must be externally wired low. In the EPROM devices, this pin also receives the programming supply voltage (V_{PP}) during EPROM programming.

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

Port 0: Port 0 is an 8-bit open drain bidirectional port. As an open drain output port, it can sink eight LS TTL loads. Port 0 pins that have 1s written to them float, and in that state will function as high impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external memory. In this application it uses strong internal pullups when emitting 1s. Port 0 emits code bytes during program verification. In this application, external pullups are required.

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, port 1 pins that are externally being pulled low will source current because of the internal pullups.

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. Port 2 emits the high-order address byte during accesses to external memory that use 16-bit addresses. In this application, it uses the strong internal pullups when emitting 1s.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. It also serves the functions of various special features of the 80C51 Family as follows:

Port Pin	Alternate Function
P3.0	RxD (serial input port)
P3.1	TxD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

V_{CC}: Supply voltage

V_{SS}: Circuit ground potential

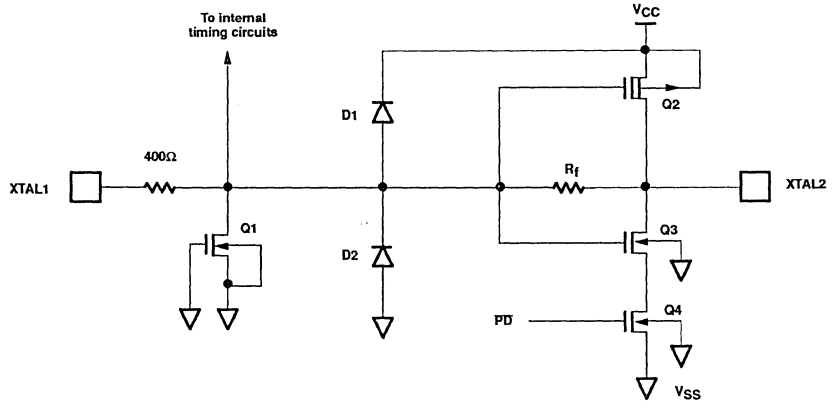


Figure 28. On-Chip Oscillator Circuitry in the CMOS Version of the 80C51 Family

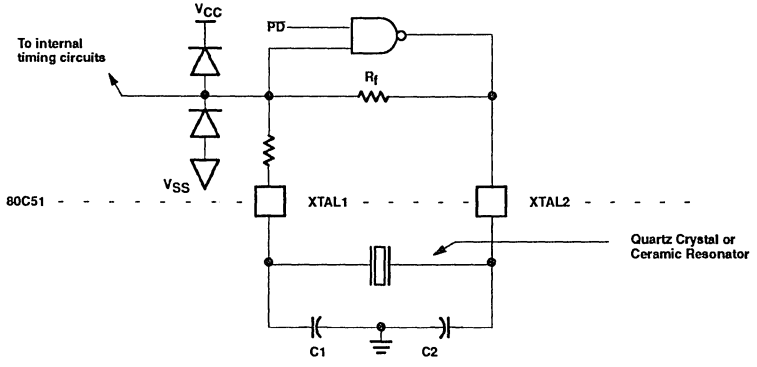


Figure 29. Using the CMOS On-Chip Oscillator

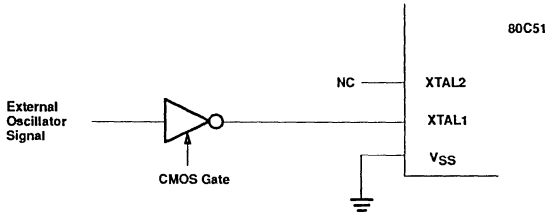


Figure 30. Driving the CMOS Family Parts with an External Clock Source

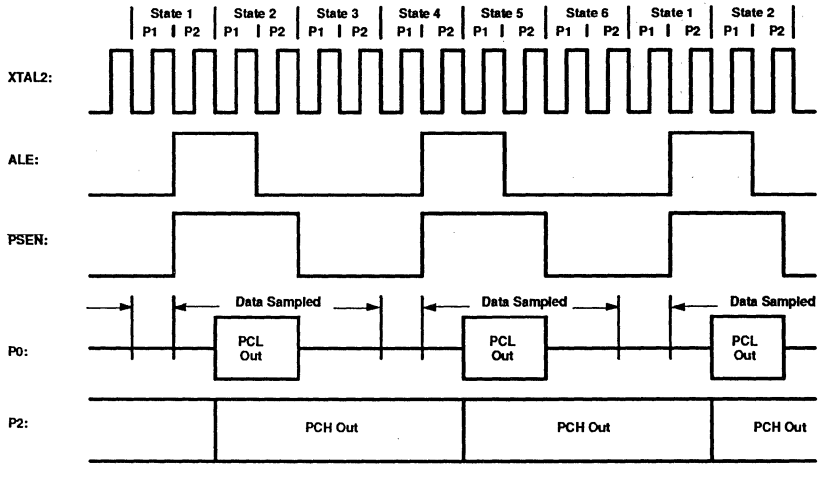


Figure 31. External Program Memory Fetches

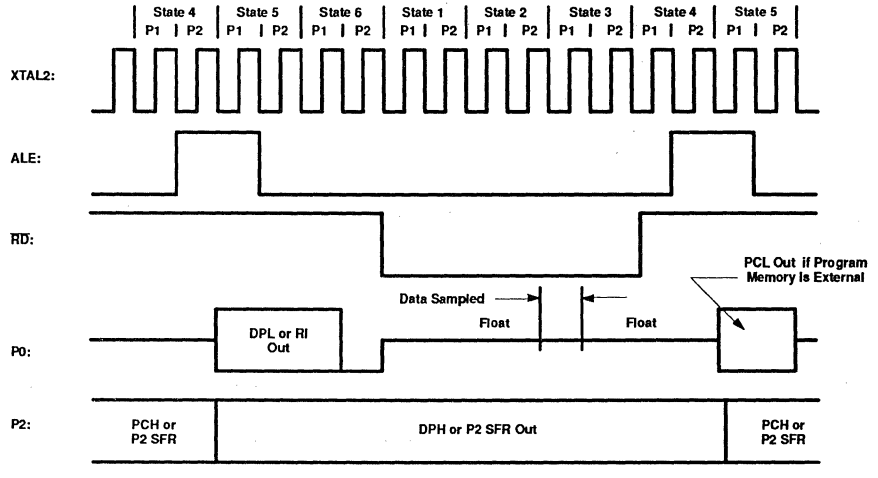


Figure 32. External Data Memory Read Cycle

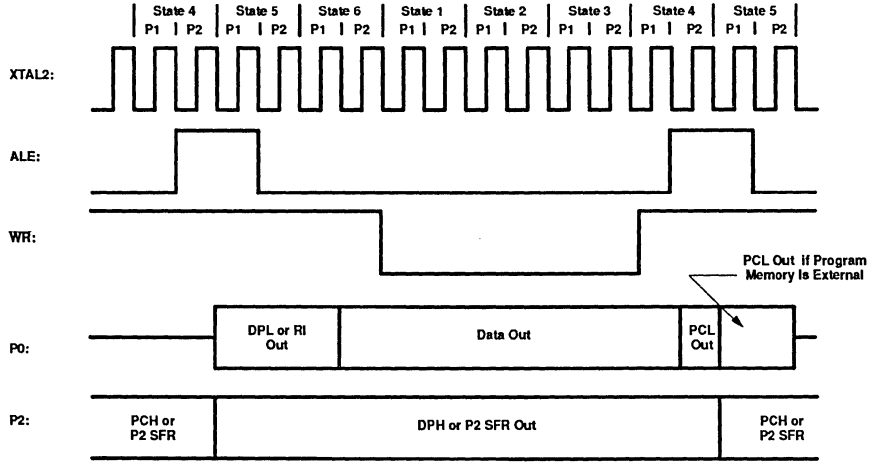


Figure 33. External Data Memory Write Cycle

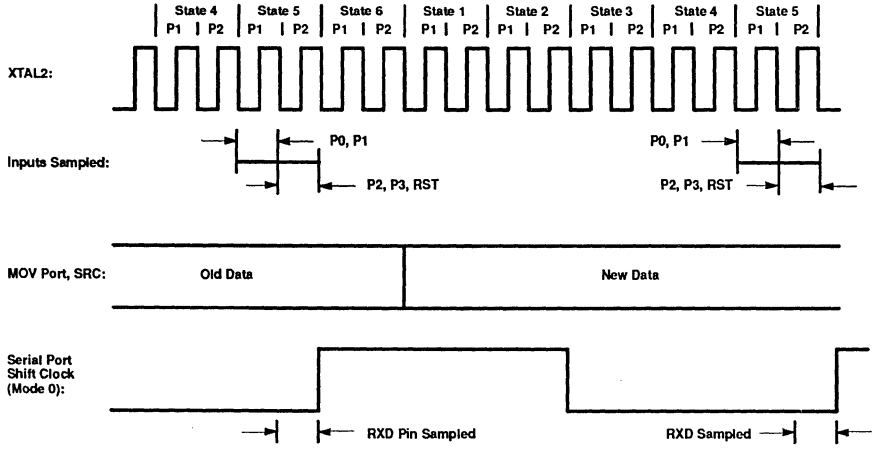


Figure 34. Port Operation

PROGRAMMER'S GUIDE AND INSTRUCTION SET

Memory Organization

Program Memory

The 80C51 has separate address spaces for program and data memory. The Program memory can be up to 64k bytes long. The lower 4k can reside on-chip. Figure 1 shows a map of the 80C51 program memory.

The 80C51 can address up to 64k bytes of data memory to the chip. The MOVX instruction is used to access the external data memory.

The 80C51 has 128 bytes of on-chip RAM, plus a number of Special Function Registers (SFRs). The lower 128 bytes of RAM can be accessed either by direct addressing (MOV data addr) or by indirect addressing (MOV @Ri). Figure 2 shows the Data Memory organization.

Direct and Indirect Address Area

The 128 bytes of RAM which can be accessed by both direct and indirect addressing can be divided into three segments as listed below and shown in Figure 3.

1. Register Banks 0-3: Locations 0 through 1FH (32 bytes). The device after reset defaults to register bank 0. To use the other register banks, the user must select them in software. Each register bank contains eight 1-byte registers 0 through 7. Reset initializes the stack pointer to location 07H, and it is incremented once to start from location 08H, which is the first register (R0) of the second register bank. Thus, in order to use more than one register bank, the SP should be initialized to a different location of the RAM where it is not used for data storage (i.e., the higher part of the RAM).
2. Bit Addressable Area: 16 bytes have been assigned for this segment, 20H-2FH. Each one of the 128 bits of this segment can be directly addressed (0-7FH). The bits can be referred to in two ways, both of which are acceptable by most assemblers. One way is to refer to their address (i.e., 0-7FH). The other way is with reference to bytes 20H to 2FH. Thus, bits 0-7 can also be referred to as bits 20.0-20.7, and bits 8-FH are the same as 21.0-21.7, and so on. Each of the 16 bytes in this segment can also be addressed as a byte.
3. Scratch Pad Area: 30H through 7FH are available to the user as data RAM. However, if the stack pointer has been initialized to this area, enough bytes should be left aside to prevent SP data destruction.

Figure 2 shows the different segments of the on-chip RAM.

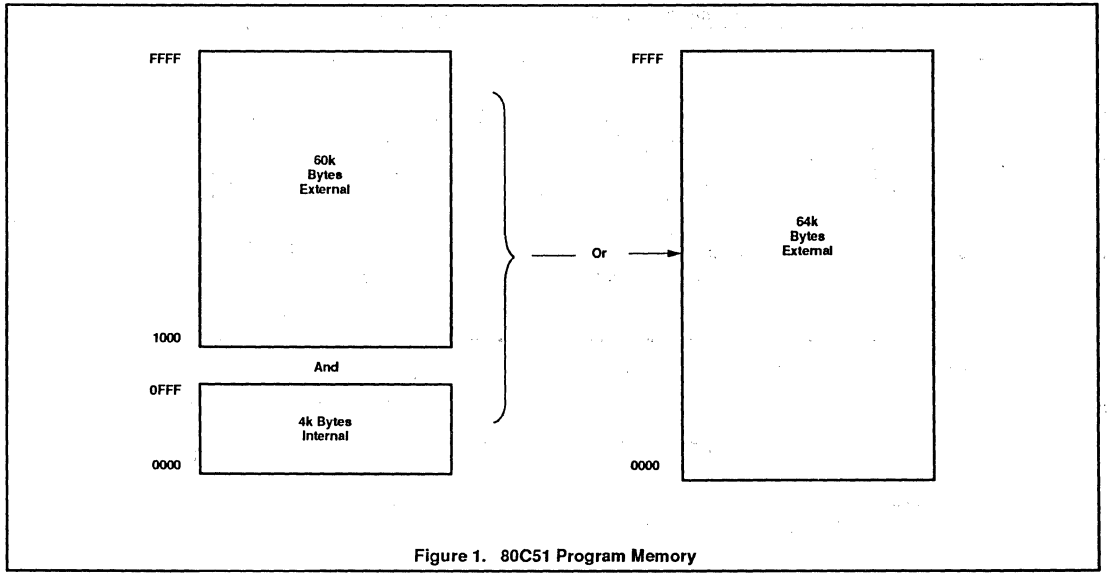


Figure 1. 80C51 Program Memory

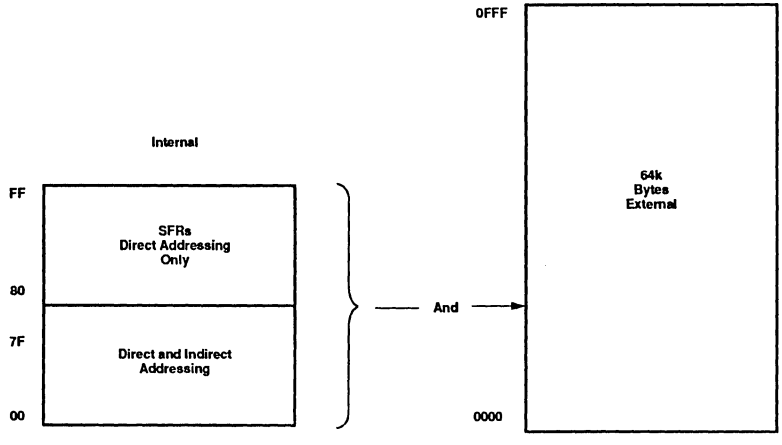


Figure 2. 80C51 Data Memory

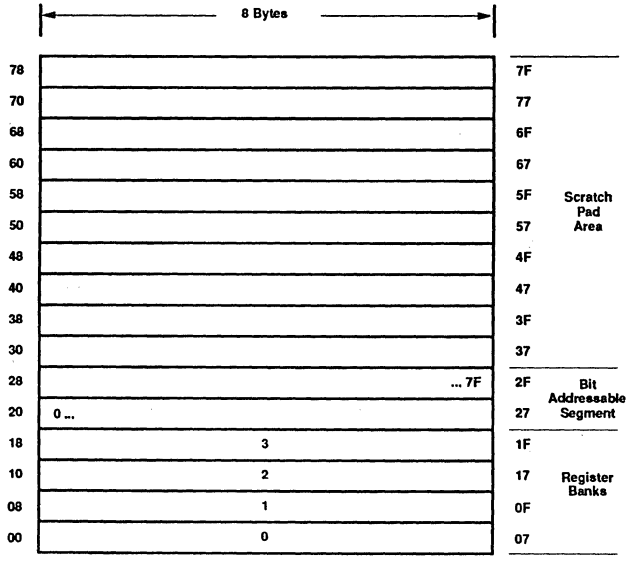


Figure 3. 128 Bytes of RAM Direct and Indirect Addressable

Table 1. 80C51 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt enable	A8H	EA	–	–	ES	ET1	EX1	ET0	EX0	0x000000B
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt priority	B8H	–	–	–	PS	PT1	PX1	PT0	PX0	xx000000B
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	–	–	–	–	–	–	T2EX	T2	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	A15	A14	A13	A12	A11	A10	A9	A8	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	$\overline{RD}$	$\overline{WR}$	T1	T0	$\overline{INT1}$	$\overline{INT0}$	TxD	Rxd	FFH
PCON[1]	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxxxxxB
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
SBUF	Serial data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial controller	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
TH0	Timer high 0	8CH									00H
TH1	Timer high 1	8DH									00H
TL0	Timer low 0	8AH									00H
TL1	Timer low 1	8BH									00H
TMOD	Timer mode	89H	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	00H

NOTES:
* Bit addressable
1. Bits GF1, GF0, PD, and IDL of the PCON register are not implemented on the NMOS 8051/8031.

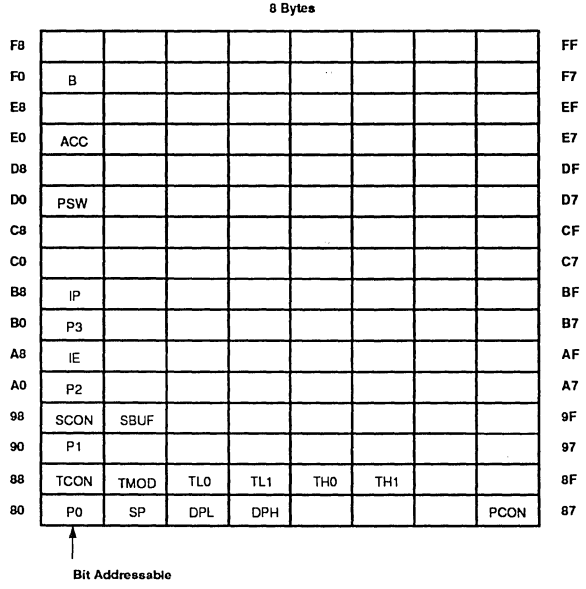


Figure 4. SFR Memory Map

Those SFRs that have their bits assigned for various functions are listed in this section. A brief description of each bit is provided for quick reference. For more detailed information refer to the Architecture Chapter of this book.

PSW: PROGRAM STATUS WORD. BIT ADDRESSABLE.

CY	AC	F0	RS1	RS0	OV	–	P

CY	PSW.7	Carry Flag.
AC	PSW.6	Auxiliary Carry Flag.
F0	PSW.5	Flag 0 available to the user for general purpose.
RS1	PSW.4	Register Bank selector bit 1 (SEE NOTE 1).
RS0	PSW.3	Register Bank selector bit 0 (SEE NOTE 1).
OV	PSW.2	Overflow Flag.
–	PSW.1	Usable as a general purpose flag.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of '1' bus in the accumulator.

NOTE:
1. The value presented by RS0 and RS1 selects the corresponding register bank.

RS1	RS0	REGISTER BANK	ADDRESS
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

PCON: POWER CONTROL REGISTER. NOT BIT ADDRESSABLE.

SMOD	–	–	–	GF1	GF0	PD	IDL

SMOD Double baud rate bit. If Timer 1 is used to generate baud rate and SMOD = 1, the baud rate is doubled when the Serial Port is used in modes 1, 2, or 3.

– Not implemented, reserved for future use.*

– Not implemented reserved for future use.*

– Not implemented reserved for future use.*

GF1 General purpose flag bit.

GF0 General purpose flag bit.

PD Power Down Bit. Setting this bit activates Power Down operation in the 80C51. (Available only in CMOS.)

IDL Idle mode bit. Setting this bit activates Idle Mode operation in the 80C51. (Available only in CMOS.)

If 1s are written to PD and IDL at the same time, PD takes precedence.

* User software should not write 1s to reserved bits. These bits may be used in future 8051 products to invoke new features.

INTERRUPTS:

To use any of the interrupts in the 80C51 Family, the following three steps must be taken.

1. **Set the EA (enable all) bit in the IE register to 1.**
2. **Set the corresponding individual interrupt enable bit in the IE register to 1.**
3. **Begin the interrupt service routine at the corresponding Vector Address of that interrupt. See Table below.**

INTERRUPT SOURCE	VECTOR ADDRESS
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI & TI	0023H

In addition, for external interrupts, pins INT0 and INT1 (P3.2 and P3.3) must be set to 1, and depending on whether the interrupt is to be level or transition activated, bits IT0 or IT1 in the TCON register may need to be set to 1.

ITx = 0 level activated

ITx = 1 transition activated

IE: INTERRUPT ENABLE REGISTER. BIT ADDRESSABLE.

If the bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

EA	–	–	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	Not implemented, reserved for future use.*
—	IE.5	Not implemented, reserved for future use.*
ES	IE.4	Enable or disable the serial port interrupt.
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE.2	Enable or disable External Interrupt 1.
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE.0	Enable or disable External Interrupt 0.

* User software should not write 1s to reserved bits. These bits may be used in future 80C51 products to invoke new features.

ASSIGNING HIGHER PRIORITY TO ONE OR MORE INTERRUPTS:

In order to assign higher priority to an interrupt the corresponding bit in the IP register must be set to 1.

Remember that while an interrupt service is in progress, it cannot be interrupted by a lower or same level interrupt.

PRIORITY WITHIN LEVEL:

Priority within level is only to resolve simultaneous requests of the same priority level.

From high to low, interrupt sources are listed below:

IE0
TF0
IE1
TF1
RI or TI

IP: INTERRUPT PRIORITY REGISTER. BIT ADDRESSABLE.

If the bit is 0, the corresponding interrupt has a lower priority and if the bit is 1 the corresponding interrupt has a higher priority.

–	–	–	PS	PT1	PX1	PT0	PX0

–	IP.7	Not implemented, reserved for future use.*
–	IP.6	Not implemented, reserved for future use.*
–	IP.5	Not implemented, reserved for future use.*
PS	IP.4	Defines the Serial Port interrupt priority level.
PT1	IP.3	Defines the Timer 1 interrupt priority level.
PX1	IP.2	Defines External Interrupt 1 priority level.
PT0	IP.1	Defines the Timer 0 interrupt priority level.
PX0	IP.0	Defines the External Interrupt 0 priority level.

* User software should not write 1s to reserved bits. These bits may be used in future 80C51 products to invoke new features.

TCON: TIMER/COUNTER CONTROL REGISTER. BIT ADDRESSABLE.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1	TCON.7	Timer 1 overflow flag. Set by hardware when the Timer/Counter 1 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
TR1	TCON.6	Timer 1 run control bit. Set/cleared by software to turn Timer/Counter 1 ON/OFF.
TF0	TCON.5	Timer 0 overflow flag. Set by hardware when the Timer/Counter 0 overflows. Cleared by hardware as processor vectors to the service routine.
TR0	TCON.4	Timer 0 run control bit. Set/cleared by software to turn Timer/Counter 0 ON/OFF.
IE1	TCON.3	External Interrupt 1 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.
IT1	TCON.2	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.
IE0	TCON.1	External Interrupt 0 edge flag. Set by hardware when External Interrupt edge detected. Cleared by hardware when interrupt is processed.
IT0	TCON.0	Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

TMOD: TIMER/COUNTER MODE CONTROL REGISTER. NOT BIT ADDRESSABLE.

GATE	C/T	M1	M0	GATE	C/T	M1	M0
		Timer 1				Timer 0	

GATE	When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).
C/T	Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).
M1	Mode selector bit. (NOTE 1)
M0	Mode selector bit. (NOTE 1)

NOTE 1:

M1	M0		Operating Mode
0	0	0	13-bit Timer (8048 compatible)
0	1	1	16-bit Timer/Counter
1	0	2	8-bit Auto-Reload Timer/Counter
1	1	3	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standart Timer 0 control bits. TH0 is an8-bit Timer and is controlled by Timer 1 control bits.
1	1	3	(Timer 1) Timer/Counter 1 stopped.

TIMER SET-UP

Tables 2 through 5 give some values for TMOD which can be used to set up Timer 0 in different modes.

It is assumed that only one timer is being used at a time. If it is desired to run Timers 0 and 1 simultaneously, in any mode, the value in TMOD for Timer 0 must be ORed with the value shown for Timer 1 (Tables 5 and 6).

For example, if it is desired to run Timer 0 in mode 1 GATE (external control), and Timer 1 in mode 2 COUNTER, then the value that must be loaded into TMOD is 69H (09H from Table 3 ORed with 60H from Table 6).

Moreover, it is assumed that the user, at this point, is not ready to turn the timers on and will do that at a different point in the program by setting bit TRx (in TCON) to 1.

TIMER/COUNTER 0

Table 2. As a Timer:

MODE	TIMER 0 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	00H	08H
1	16-bit Timer	01H	09H
2	8-bit Auto-Reload	02H	0AH
3	Two 8-bit Timers	03H	0BH

Table 3. As a Counter:

MODE	COUNTER 0 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	04H	0CH
1	16-bit Timer	05H	0DH
2	8-bit Auto-Reload	06H	0EH
3	One 8-bit Counter	07H	0FH

NOTES:
1. The timer is turned ON/OFF by setting/clearing bit TR0 in the software.
2. The Timer is turned ON/OFF by the 1-to-0 transition on $\overline{INT0}$ (P3.2) when TR0 = 1 (hardware control).

TIMER/COUNTER 1

Table 4. As a Timer:

MODE	TIMER 1 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	00H	80H
1	16-bit Timer	10H	90H
2	8-bit Auto-Reload	20H	A0H
3	Does not run	30H	B0H

Table 5. As a Counter:

MODE	COUNTER 1 FUNCTION	TMOD	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	40H	C0H
1	16-bit Timer	50H	D0H
2	8-bit Auto-Reload	60H	E0H
3	Not available	–	–

NOTES:
1. The timer is turned ON/OFF by setting/clearing bit TR1 in the software.
2. The Timer is turned ON/OFF by the 1-to-0 transition on $\overline{INT1}$ (P3.2) when TR1 = 1 (hardware control).

SCON: SERIAL PORT CONTROL REGISTER. BIT ADDRESSABLE.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SCON.7	Serial Port mode specifier. (NOTE 1)
SM1	SCON.6	Serial Port mode specifier. (NOTE 1)
SM2	SCON.5	Enables the multiprocessor communication feature in modes 2 & 3. In mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0. (See Table 6.)
REN	SCON.4	Set/Cleared by software to Enable/Disable reception.
TB8	SCON.3	The 9th bit that will be transmitted in modes 2 & 3. Set/Cleared by software.
RB8	SCON.2	In modes 2 & 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.
TI	SCON.1	Transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes. Must be cleared by software.
RI	SCON.0	Receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or halfway through the stop bit time in the other modes (except see SM2). Must be cleared by software.

NOTE 1:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	Shift Register	$F_{OSC.}/12$
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	$F_{OSC.}/64$ or $F_{OSC.}/32$
1	1	3	9-bit UART	Variable

SERIAL PORT SET-UP:

Table 6.

MODE	SCON	SM2 VARIATION
0 1 2 3	10H 50H 90H D0H	Single Processor Environment (SM2 = 0)
0 1 2 3	NA 70H B0H F0H	Multiprocessor Environment (SM2 = 1)

GENERATING BAUD RATES

Serial Port in Mode 0:

Mode 0 has a fixed baud rate which is 1/12 of the oscillator frequency. To run the serial port in this mode none of the Timer/Counters need to be set up. Only the SCON register needs to be defined.

$$\text{Baud Rate} = \frac{\text{Osc Freq}}{12}$$

Serial Port in Mode 1:

Mode 1 has a variable baud rate. The baud rate is generated by Timer 1.

USING TIMER/COUNTER 1 TO GENERATE BAUD RATES:

For this purpose, Timer 1 is used in mode 2 (Auto-Reload). Refer to Timer Setup section of this chapter.

$$\text{Baud Rate} = \frac{K \times \text{Osc Freq}}{32 \times 12 \times [256 - (TH1)]}$$

If SMOD = 0, then K = 1.
If SMOD = 1, then K = 2 (SMOD is in the PCON register).

Most of the time the user knows the baud rate and needs to know the reload value for TH1.

$$TH1 = 256 - \frac{K \times \text{Osc Freq}}{384 \times \text{baud rate}}$$

TH1 must be an integer value. Rounding off TH1 to the nearest integer may not produce the desired baud rate. In this case, the user may have to choose another crystal frequency.

Since the PCON register is not bit addressable, one way to set the bit is logical ORing the PCON register (i.e., ORL PCON,#80H). The address of PCON is 87H.

SERIAL PORT IN MODE 2:

The baud rate is fixed in this mode and is 1/32 or 1/64 of the oscillator frequency, depending on the value of the SMOD bit in the PCON register.

In this mode none of the Timers are used and the clock comes from the internal phase 2 clock.

SMOD = 1, Baud Rate = 1/32 Osc Freq.

SMOD = 0, Baude Rate = 1/64 Osc Freq.

To set the SMOD bit: ORL PCON,#80H. The address of PCON is 87H.

SERIAL PORT IN MODE 3:

The baud rate in mode 3 is variable and sets up exactly the same as in mode 1.

80C51 FAMILY INSTRUCTION SET

Table 7. 80C51 Instruction Set Summary

Interrupt Response Time: Refer to Hardware Description Chapter.

Instructions that Affect Flag Settings[1]

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ANL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

[1]Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Notes on instruction set and addressing modes:

Rn	Register R7-R0 of the currently selected Register Bank.
direct	8-bit internal data location's address. This could be an Internal Data RAM location (0-127) or a SFR [i.e., I/O port, control register, status register, etc. (128-255)].
@Ri	8-bit internal data RAM location (0-255) addressed indirectly through register R1 or R0.
#data	8-bit constant included in the instruction.
#data 16	16-bit constant included in the instruction
addr 16	16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64k-byte Program Memory address space.
addr 11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2k-byte page of program memory as the first byte of the following instruction.
rel	Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is –128 to +127 bytes relative to first byte of the following instruction.
bit	Direct Addressed bit in Internal Data RAM or Special Function Register.

MNEMONIC		DESCRIPTION	BYTE	OSCILLATOR PERIOD
ARITHMETIC OPERATIONS				
ADD	A,Rn	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@Ri	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,Rn	Add register to Accumulator with carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with carry	2	12
ADDC	A,@Ri	Add indirect RAM to Accumulator with carry	1	12
ADDC	A,#data	Add immediate data to A_{CC} with carry	2	12
SUBB	A,Rn	Subtract Register from A_{CC} with borrow	1	12
SUBB	A,direct	Subtract direct byte from A_{CC} with borrow	2	12
SUBB	A,@Ri	Subtract indirect RAM from A_{CC} with borrow	1	12
SUBB	A,#data	Subtract immediate data from A_{CC} with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	Rn	Increment register	1	12

All mnemonics copyrighted © Intel Corporation 1980

Table 7. 80C51 Instruction Set Summary (Continued)

MNEMONIC		DESCRIPTION	BYTE	OSCILLATOR PERIOD
ARITHMETIC OPERATIONS (Continued)				
INC	direct	Increment direct byte	2	12
INC	@Ri	Increment indirect RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	Rn	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@Ri	Decrement indirect RAM	1	12
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A and B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12
LOGICAL OPERATIONS				
ANL	A,Rn	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@Ri	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,Rn	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@Ri	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,Rn	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12
RL	A	Rotate Accumulator left	1	12
RLC	A	Rotate Accumulator left through the carry	1	12
RR	A	Rotate Accumulator right	1	12
RRC	A	Rotate Accumulator right through the carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,Rn	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@Ri	Move indirect RAM to Accumulator	1	12

All mnemonics copyrighted © Intel Corporation 1980

Table 7. 80C51 Instruction Set Summary (Continued)

MNEMONIC		DESCRIPTION	BYTE	OSCILLATOR PERIOD
DATA TRANSFER (Continued)				
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	Rn,A	Move Accumulator to register	1	12
MOV	Rn,direct	Move direct byte to register	2	24
MOV	RN,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,Rn	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@Ri	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@Ri,A	Move Accumulator to indirect RAM	1	12
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to A_{CC}	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to A_{CC}	1	24
MOVX	A,@Ri	Move external RAM (8-bit addr) to A_{CC}	1	24
MOVX	A@DPTR	Move external RAM (16-bit addr) to A_{CC}	1	24
MOVX	A,@Ri,A	Move A_{CC} to external RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move A_{CC} to external RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order digit indirect RAM with A_{CC}	1	12
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to carry	2	24
ANL	C,/bit	AND complement of direct bit to carry	2	24
ORL	C,bit	OR direct bit to carry	2	24
ORL	C,/bit	OR complement of direct bit to carry	2	24
MOV	C,bit	Move direct bit to carry	2	12
MOV	bit,C	Move carry to direct bit	2	24
JC	rel	Jump if carry is set	2	24
JNC	rel	Jump if carry not set	2	24

All mnemonics copyrighted © Intel Corporation 1980

Table 7. 80C51 Instruction Set Summary (Continued)

MNEMONIC		DESCRIPTION	BYTE	OSCILLATOR PERIOD
BOOLEAN VARIABLE MANIPULATION (Continued)				
JB	rel	Jump if direct bit is set	2	24
JNB	rel	Jump if direct bit is not set	2	24
JBC	bit,rel	Jump if direct bit is set and clear bit	3	24
PROGRAM BRANCHING				
ACALL	addr11	Absolute subroutine call	2	24
LCALL	addr16	Long subroutine call	3	24
RET		Return from subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute jump	2	24
LJMP	addr16	Long jump	3	24
SJMP	rel	Short jump (relative addr)	2	24
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is zero	2	24
JNZ	rel	Jump if Accumulator is not zero	2	24
CJNE	A,direct,rel	Compare direct byte to A_{CC} and jump if not equal	3	24
CJNE	A,#data,rel	Compare immediate to A_{CC} and jump if not equal	3	24
CJNE	RN,#data,rel	Compare immediate to register and jump if not equal	3	24
CJNE	@Ri,#data,rel	Compare immediate to indirect and jump if not equal	3	24
DJNZ	Rn,rel	Decrement register and jump if not zero	2	24
DJNZ	direct,rel	Decrement direct byte and jump if not zero	3	24
NOP		No operation	1	12

All mnemonics copyrighted © Intel Corporation 1980

INSTRUCTION DEFINITIONS

ACALL addr11

Function: Absolute Call

Description: ACALL unconditionally calls a subroutine located at the indicated address. The instruction increments the PC twice to obtain the address of the following instruction, then pushes the 16-bit result onto the stack (low-order byte first) and increments the Stack Pointer twice. The destination address is obtained by successively concatenating the five high-order bits of the incremented PC, opcode bits 7-5, and the second byte of the instruction. The subroutine called must therefore start within the same 2k block of the program memory as the first byte of the instruction following ACALL. No flags are affected.

Example: Initially SP equals 07H. The label "SUBRTN" is at program memory location 0345 H. After executing the instruction,

ACALL SUBRTN

at location 0123H, SP will contain 09H, internal RAM locations 08H and 09H will contain 25H and 01H, respectively, and the PC will contain 0345H.

Bytes: 2

Cycles: 2

Encoding:

a10 a9 a8 1	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0

Operation: ACALL
(PC) ← (PC) + 2
(SP) ← (SP) + 1
(SP) ← (PC_{7-0})
(SP) ← (SP) + 1
(SP) ← (PC_{15-8})
(PC_{10-0}) ← page address

ADD A,<src-byte>

Function: Add

Description: ADD adds the byte variable indicated to the Accumulator, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B). The instruction,

ADD A,R0

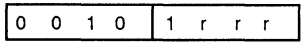
will leave 6DH (01101101B) in the Accumulator with the AC flag cleared and both the Carry flag and OV set to 1.

ADD A,Rn

Bytes: 1

Cycles: 1

Encoding:



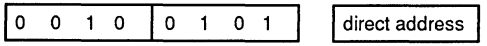
Operation: ADD
(A) ← (A) + (R_n)

ADD A,direct

Bytes: 2

Cycles: 1

Encoding:



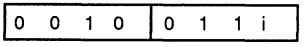
Operation: ADD
(A) ← (A) + (direct)

ADD A,@Ri

Bytes: 1

Cycles: 1

Encoding:



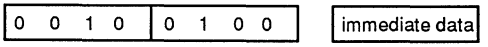
Operation: ADD
(A) ← (A) + ((R_i))

ADD A,#data

Bytes: 2

Cycles: 1

Encoding:



Operation: ADD
(A) ← (A) + #data

ADDC A,<src-byte>

Function: Add with Carry

Description: ADDC simultaneously adds the byte variable indicated, the carry flag and the Accumulator contents, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not out of bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) with the carry flag set. The instruction,

ADDC A,R0

will leave 6EH (01101110B) in the Accumulator with AC cleared and both the Carry flag and OV set to 1.

ADDC A,Rn

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	1	r	r	r

Operation: ADDC
(A) ← (A) + (C) + (R_n)

ADDC A,direct

Bytes: 2

Cycles: 1

Encoding:

0	0	1	1	0	1	0	1	direct address

Operation: ADDC
(A) ← (A) + (C) + (direct)

ADDC A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	0	1	1	i

Operation: ADDC
(A) ← (A) + (C) + ((R_i))

ADDC A,#data

Bytes: 2

Cycles: 1

Encoding:

0	0	1	1	0	1	0	0	immediate data

Operation: ADDC
(A) ← (A) + (C) + #data

AJMP addr11

Function: Absolute Jump

Description: AJMP transfers program execution to the indicated address, which is formed at run-time by concatenating the high-order five bits of the PC (*after* incrementing the PC twice), opcode bits 7-5, and the second byte of the instruction. The destination must therefore be within the same 2k block of program memory as the first byte of the instruction following AJMP.

Example: The label "JMPADR" is at program memory location 0123H. The instruction,

AJMP JMPADR

is at location 0345H and will load the PC with 0123H.

Bytes: 2

Cycles: 2

Encoding:

a10 a9 a8 0	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0

Operation: AJMP
(PC) ← (PC) + 2
($PC_{10\text{-}0}$) ← page address

ANL <dest-byte>,<src-byte>

Function: Logical-AND for byte variables

Description: ANL performs the bitwise logical-AND operation between the variables indicated and stores the results in the destination variable. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: If the Accumulator holds 0C3H (11000011B) and register 0 holds 55H (01010101B) then the instruction,

ANL A,R0

will leave 41H (01000001B) in the Accumulator.

When the destination is a directly addressed byte, this instruction will clear combinations of bits in any RAM location or hardware register. The mask byte determining the pattern of bits to be cleared would either be a constant contained in the instruction or a value computed in the Accumulator at run-time. The instruction,

ANL P1,#0111001B

will clear bits 7, 3, and 2 of output port 1.

ANL A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 1 0 1	1 r r r

Operation: ANL
(A) ← (A) ∧ (R_n)

ANL A,direct

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1	0 1 0 1	direct address

Operation: ANL
(A) ← (A) ∧ (direct)

ANL A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 1 0 1	0 1 1 i

Operation: ANL
(A) ← (A) ∧ ((R_i))

ANL A,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1	0 1 0 0	immediate data

Operation: ANL
(A) ← (A) ∧ #data

ANL direct,A

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1	0 0 1 0	direct address

Operation: ANL
(A) ←(direct) ∧ (A)

ANL direct,#data

Bytes: 3

Cycles: 2

Encoding:

0 1 0 1	0 0 1 1	direct address	immediate data

Operation: ANL
(direct) ←(direct) ∧ #data

ANL C,<src-bit>

Function: Logical-AND for bit variables

Description: If the Boolean value of the source bit is a logical 0 then clear the carry flag; otherwise leave the carry flag in its current state. A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, *but the source bit itself is not affected.* No other flags are affected.

Only direct addressing is allowed for the source operand.

Example: Set the carry flag if, and only if, P1.0 = 1, ACC.7 = 1, and OV = 0:

```
MOV   C,P1.0   ;LOAD CARRY WITH INPUT PIN STATE
ANL   C,ACC.7  ;AND CARRY WITH ACCUM. BIT 7
ANL   C,/OV    ;AND WITH INVERSE OF OVERFLOW FLAG
```

ANL C,bit

Bytes: 2

Cycles: 2

Encoding:

1 0 0 0	0 0 1 0	bit address

Operation: ANL

$(C) \leftarrow (C) \wedge (bit)$

ANL C,/bit

Bytes: 2

Cycles: 2

Encoding:

1 0 1 1	0 0 0 0	bit address

Operation: ANL

$(C) \leftarrow (C) \wedge \overline{(bit)}$

CJNE <dest-byte>,<src-byte>,rel

Function: Compare and Jump if Not Equal

Description: CJNE compares the magnitudes of the first two operands, and branches if their values are not equal. The branch destination is computed by adding the signed relative-displacement in the last instruction byte to the PC, after incrementing the PC to the start of the next instruction. The carry flag is set if the unsigned integer value of <dest-byte> is less than the unsigned integer value of <src-byte>; otherwise, the carry is cleared. Neither operand is affected.

The first two operands allow four addressing mode combinations: the Accumulator may be compared with any directly addressed byte or immediate data, and any indirect RAM location or working register can be compared with an immediate constant.

Example: The Accumulator contains 34H. Register 7 contains 56H. The first instruction in the sequence,

```
        CJNE    R7,#60H,NOT_EQ
;       ...     ....        ;       R7 = 60H.
NOT_EQ  JC      REQ_LOW     ;       IF R7 < 60H.
;       ...     ....        ;       R7 > 60H.
```

sets the carry flag and branches to the instruction at label NOT_EQ. By testing the carry flag, this instruction determines whether R7 is greater or less than 60H.

If the data being presented to Port 1 is also 34H, then the instruction,

WAIT: CJNE A,P1,WAIT

clears the carry flag and continues with the next instruction in sequence, since the Accumulator does equal the data read from P1. (If some other value was being input on P1, the program will loop at this point until the P1 data changes to 34H.)

CJNE A,direct,rel

Bytes: 3

Cycles: 2

Encoding:

1 0 1 1	0 1 0 1	direct address	rel. address

Operation:

(PC) ← (PC) + 3
IF (A) < > *(direct)*
THEN
(PC) ← (PC) + *relative offset*
IF (A) < *(direct)*
THEN.
(C) ← 1
ELSE
(C) ← 0

CJNE A,#data,rel

Bytes: 3

Cycles: 2

Encoding:

1 0 1 1	0 1 0 0	immediate data	rel. address

Operation:

(PC) ← (PC) + 3
IF (A) < > *data*
THEN
(PC) ← (PC) + *relative offset*
IF (A) < *data*
THEN
(C) ← 1
ELSE
(C) ← 0

CJNE Rn,#data,rel

Bytes: 3

Cycles: 2

Encoding:

1 0 1 1	1 r r r	immediate data	rel. address

Operation:

(PC) ← (PC) + 3
IF (R_n) < > *data*
THEN
(PC) ← (PC) + *relative offset*
IF (R_n) < *data*
THEN
(C) ← 1
ELSE
(C) ← 0

CJNE @Ri,#data,rel

Bytes: 3

Cycles: 2

Encoding:

1 0 1 1	0 1 1 i	immediate data	rel. address

Operation:

(PC) ← (PC) + 3
IF ((R_i)) < > *data*
THEN
(PC) ← (PC) + *relative offset*
IF ((R_i)) < *data*
THEN
(C) ← 1
ELSE
(C) ← 0

CLR A

Function: Clear Accumulator

Description: The Accumulator is cleared (all bits reset to zero). No flags are affected.

Example: The Accumulator contains 5CH (01011100B). The instruction,

CLR A

will leave the Accumulator set to 00H (00000000B).

Bytes: 1

Cycles: 1

Encoding:

1 1 1 0	0 1 0 0

Operation:

CLR
(A) ← 0

CLR bit

Function: Clear bit

Description: The indicated bit is cleared (reset to zero). No other flags are affected. CLR can operate on the carry flag or any directly addressable bit.

Example: Port 1 has previously been written with 5DH (01011101B). The instruction,

CLR P1.2

will leave the port set to 59H (01011001B).

CLR C

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0	0 0 1 1

Operation:

CLR
(C) ← 0

CLR bit

Bytes: 2

Cycles: 1

Encoding:

1 1 0 0	0 0 1 0	bit address

Operation:

CLR
(bit) ← 0

CPL A

Function: Complement Accumulator

Description: Each bit of the Accumulator is logically complemented (one's complement). Bits which previously contained a one are changed to a zero and vice-versa. No flags are affected.

Example: The Accumulator contains 5CH (01011100B). The instruction,

CPL A

will leave the Accumulator set to 0A3H (10100011B).

Bytes: 1

Cycles: 1

Encoding:

1 1 1 1	0 1 0 0

Operation: CPL
(A) ← ⌉ (A)

CPL bit

Function: Complement bit

Description: The bit variable specified is complemented. A bit which had been a one is changed to zero and vice-versa. No other flags are affected. CLR can operate on the carry or any directly addressable bit.

Note: When this instruction is used to modify an output pin, the value used as the original data will be read from the output data latch, *not* the input pin.

Example: Port 1 has previously been written with 5DH (01011101B). The instruction sequence,

CPL P1.1

CPL P1.2

will leave the port set to 5BH (01011011B).

CPL C

Bytes: 1

Cycles: 1

Encoding:

1 0 1 1	0 0 1 1

Operation: CPL
(C) ← ⌉ (C)

CPL bit

Bytes: 2

Cycles: 1

Encoding:

1 0 1 1	0 0 1 0	bit address

Operation: CPL
(bit) ← ⌉ (bit)

DA A

Function: Decimal-adjust Accumulator for Addition

Description: DA A adjusts the eight-bit value in the Accumulator resulting from the earlier addition of two variable (each in packed-BCD format), producing two four-bit digits. Any ADD or ADDC instruction may have been used to perform the addition.

If Accumulator bits 3-0 are greater than nine (xxx1010-xxx1111), or if the AC flag is one, six is added to the Accumulator, producing the proper BCD digit in the low-order nibble. This internal addition would set the carry flag if a carry-out of the low-order four-bit field propagated through all high-order bits, but it would not clear the carry flag otherwise.

If the carry flag is now set, or if the four high-order bits now exceed nine (1010xxx-111xxxx), these high-order bits are incremented by six, producing the proper BCD digit in the high-order nibble. Again, this would set the carry flag if there was a carry-out of the high-order bits, but wouldn't clear the carry. The carry flag thus indicates if the sum of the original two BCD variables is greater than 100, allowing multiple precision decimal addition. OV is not affected.

All of this occurs during the one instruction cycle. Essentially, this instruction performs the decimal conversion by adding 00H, 06H, 60H, or 66H to the Accumulator, depending on initial Accumulator and PSW conditions.

Note: DA A *cannot* simply convert a hexadecimal number in the Accumulator to BCD notation, nor does DA A apply to decimal subtraction.

Example: The Accumulator holds the value 56H (01010110B) representing the packed BCD digits of the decimal number 56. Register 3 contains the value 67H (01100111B) representing the packed BCD digits of the decimal number 67. The carry flag is set.. The instruction sequence,

```
ADDC    A,R3
DA      A
```

will first perform a standard two's-complement binary addition, resulting in the value 0BEH (10111110B) in the Accumulator. The carry and auxiliary carry flags will be cleared.

The Decimal Adjust instruction will then alter the Accumulator to the value 24H (00100100B), indicating the packed BCD digits of the decimal number 24, the low-order two digits of the decimal sum of 56, 67, and the carry-in. The carry flag will be set by the Decimal Adjust instruction, indicating that a decimal overflow occurred. The true sum 56, 67, and 1 is 124.

BCD variables can be incremented or decremented by adding 01H or 99H. If the Accumulator initially holds 30H (representing the digits of 30 decimal), the the instruction sequence,

```
ADD     A,#99H
DA      A
```

will leave the carry set and 29H in the Accumulator, since 30 + 99 = 129. The low-order byte of the sum can be interpreted to mean 30 – 1 = 29.

Bytes: 1

Cycles: 1

Encoding:

1	1	0	1	0	1	0	0

Operation:

```
DA
–contents of Accumulator are BCD
IF   [[(A3-0) > 9] ∨ [(AC) = 1]]
     THEN(A3-0) ← (A3-0) + 6
          AND
IF   [[(A7-4) > 9] ∨ [(C) = 1]]
     THEN(A7-4) ← (A7-4) + 6
```

DEC byte

Function: Decrement

Description: The variable indicated is decremented by 1. An original value of 00H will underflow to 0FFH. No flags are affected. Four operand addressing modes are allowed: accumulator, register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original data will be read from the output data latch, *not* the input pin.

Example: Register 0 contains 7FH (01111111B). Internal RAM locations 7EH and 7FH contain 00H and 40H, respectively. The instruction sequence,

```
DEC   @R0
DEC   R0
DEC   @R0
```

will leave register 0 set to 7EH and internal RAM locations 7EH and 7FH set to 0FFH and 3FH.

DEC A

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	0 1 0 0

Operation: DEC
$(A) \leftarrow (A) - 1$

DEC Rn

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	1 r r r

Operation: DEC
$(R_n) \leftarrow (R_n) - 1$

DEC direct

Bytes: 2

Cycles: 1

Encoding:

0 0 0 1	0 1 0 1	direct address

Operation: DEC
(direct) ← (direct) – 1

DEC @Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	0 1 1 i

Operation: DEC
$((R_i)) \leftarrow ((R_i)) - 1$

DIV AB

Function: Divide

Description: DIV AB divides the unsigned eight-bit integer in the Accumulator by the unsigned eight-bit integer in register B.

The Accumulator receives the integer part of the quotient; register B receives the integer remainder. The carry and OV flags will be cleared.

Exception: if B had originally contained 00H, the values returned in the Accumulator and B-register will be undefined and the overflow flag will be set. The carry flag is cleared in any case.

Example: The Accumulator contains 251 (0FBH or 11111011B) and B contains 18 (12H or 00010010B). The instruction,

DIV AB

will leave 13 in the Accumulator (0DH or 00001101B) and the value 17 (11H or 00010001B) in B, since 251 = (13 x 18) + 17. Carry and OV will both be cleared.

Bytes: 1

Cycles: 4

Encoding:

1	0	0	0	0	1	0	0

Operation: DIV

$(A)_{15-8} \leftarrow (A)/(B)$

$(B)_{7-0}$

DJNZ <byte>,<rel-addr>

Function: Decrement and Jump if Not Zero

Description: DJNZ decrements the location indicated by 1, and branches to the address indicated by the second operand if the resulting value is not zero. An original value of 00H will underflow to 0FFH. No flags are affected. The branch destination would be computed by adding the signed relative-displacement value in the last instruction byte to the PC, after incrementing the PC to the first byte of the following instruction.

The location decremented may be a register or directly addressed byte.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Internal RAM locations 40H, 50H, and 60H contain the values 01H, 70H, and 15H, respectively. The instruction sequence,

```
DJNZ    40H,LABEL_1
DJNZ    50H,LABEL_2
DJNZ    60H,LABEL_3
```

will cause a jump to the instruction at LABEL_2 with the values 00h, 6FH, and 15H in the three RAM locations. The first jump was *not* taken because the result was zero.

This instruction provides a simple was of executing a program loop a given number of times, or for adding a moderate time delay (from 2 to 512 machine cycles) with a single instruction. The instruction sequence,

```
        MOV     R2,#8
TOGGLE: CPL     P1.7
        DJNZ    R2,TOGGLE
```

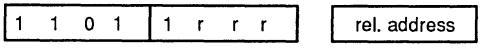
will toggle P1.7 eight times, causing four output pulses to appear at bit 7 of output Port 1. Each pulse will last three machine cycles, two for DJNZ and one to alter the pin.

DJNZ Rn,rel

Bytes: 2

Cycles: 2

Encoding:



Operation:

```
DJNZ
(PC) ← (PC) + 2
(Rn) ← (Rn) – 1
IF (Rn) > 0 or (Rn) < 0
   THEN
         (PC) ← (PC) + rel
```

DJNZ direct,rel

Bytes: 3

Cycles: 2

Encoding:

Operation:

```
DJNZ
(PC) ← (PC) + 2
(direct) ← (direct) – 1
IF (direct) > 0 or (direct) < 0
   THEN
        (PC) ← (PC) + rel
```

INC <byte>

Function: Increment

Description: INC increments the indicated variable by 1. An original value of 0FFH will overflow to 00H. No flags are affected. Three addressing modes are allowed: register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: Register 0 contains 7EH (01111110B). Internal RAM locations 7EH and 7FH contain 0FFH and 40H, respectively. The instruction sequence,

```
INC     @R0
INC     R0
INC     @R0
```

will leave register 0 set to 7FH and internal RAM locations 7EH and 7FH holding (respectively) 00H and 41H.

INC A

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	0 1 0 0

Operation: INC
(A) ← (A) + 1

INC Rn

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	1 r r r

Operation: INC
(R_n) ← (R_n) + 1

INC direct

Bytes: 2

Cycles: 1

Encoding:

0 0 0 0	0 1 0 1	direct address

Operation: INC
(direct) ← (direct) + 1

INC @Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	0 1 1 i

Operation: INC
((R_i)) ← ((R_i)) + 1

INC DPTR

Function: Increment Data Pointer

Description: Increment the 16-bit data pointer by 1. A 16-bit increment (modulo 2^{16}) is performed; an overflow of the low-order byte of the data pointer (DPL) from 0FFH to 00H will increment the high-order byte (DPH). No flags are affected.

This is the only 16-bit register which can be incremented.

Example: Registers DPH and DPL contain 12H and 0FEH, respectively. The instruction sequence,

```
INC     DPTR
INC     DPTR
INC     DPTR
```

will change DPH and DPL to 13H and 01H.

Bytes: 1

Cycles: 2

Encoding:

1 0 1 0	0 0 1 1

Operation: INC
(DPTR) ← (DPTR) + 1

JB bit,rel

Function: Jump if Bit set

Description: If the indicated bit is a one, jump to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

Example: The data present at input port 1 is 11001010B. The Accumulator holds 56 (01010110B). The instruction sequence,

```
JB      P1.2,LABEL1
JB      ACC.2,LABEL2
```

will cause program execution to branch to the instruction at label LABEL2.

Bytes: 3

Cycles: 2

Encoding:

0 0 1 0	0 0 0 0	bit address	rel. address

Operation: JB
(PC) ← (PC) + 3
IF (bit) = 1
THEN
(PC) ← (PC) + rel

JBC bit,rel

Function: Jump if Bit is set and Clear bit

Description: If the indicated bit is a one, branch to the address indicated; otherwise proceed with the next instruction. *The bit will not be cleared if it is already a zero.* The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. No flags are affected.

Note: When this instruction is used to test an output pin, the value used as the original data will read from the output data latch, not the input pin.

Example: The Accumulator holds 56H (01010110B). The instruction sequence,

```
JBC     ACC.3,LABEL1
JBC     ACC.2,LABEL2
```

will cause program execution to continue at the instruction identified by the LABEL2, with the Accumulator modified to 52H (01010010B).

Bytes: 3

Cycles: 2

Encoding:

0 0 0 1	0 0 0 0	bit address	rel. address

Operation:

```
JBC
(PC) ← (PC) + 3
IF (bit) = 1
   THEN
(bit) ← 0
(PC) ← (PC) + rel
```

JC rel

Function: Jump if Carry is set

Description: If the carry flag is set, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. No flags are affected.

Example: The carry flag is cleared. The instruction sequence,

```
JC      LABEL1
CPL     C
JC      LABEL2
```

will set the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0 1 0 0	0 0 0 0	rel. address

Operation:

```
JC
(PC) ← (PC) + 2
IF (C) = 1
   THEN
            (PC) ← (PC) + rel
```

JMP @A+DPTR

Function: Jump indirect

Description: Add the eight-bit unsigned contents of the Accumulator with the sixteen-bit data pointer, and load the resulting sum to the program counter. This will be the address for subsequent instruction fetches. Sixteen-bit addition is performed (modulo 2^{16}): a carry-out from the low-order eight bits propagates through the higher-order bits. Neither the Accumulator nor the Data Pointer is altered. No flags are affected.

Example: An even number from 0 to 6 is in the Accumulator. The following sequence of instructions will branch to one of four AJMP instructions in a jump table starting at JMP_TBL:

```
            MOV   DPTR,#JMP_TBL
            JMP   @A+DPTR
JMP_TBL:    AJMP  LABEL0
            AJMP  LABEL1
            AJMP  LABEL2
            AJMP  LABEL3
```

If the Accumulator equals 04H when starting this sequence, execution will jump to label LABEL2. Remember that AJMP is a two-byte instruction, so the jump instructions start at every other address.

Bytes: 1

Cycles: 2

Encoding:

0	1	1	1	0	0	1	1

Operation: JMP
(PC) ← (A) + (DPTR)

JNB bit,rel

Function: Jump if Bit Not set

Description: If the indicated bit is a zero, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

Example: The data present at input port 1 is 11001010B. The Accumulator holds 56H (01010110B). The instruction sequence,

```
JNB     P1.3,LABEL1
JNB     ACC.3,LABEL2
```

will cause program execution to continue at the instruction at label LABEL2.

Bytes: 3

Cycles: 2

Encoding:

0	0	1	1	0	0	0	0	bit address	rel. address

Operation: JNB
(PC) ← (PC) + 3
IF (bit) = 0
THEN
(PC) ← (PC) + rel

JNC rel

Function: Jump if Carry Not set

Description: If the carry flag is a zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice to point to the next instruction. The carry flag is not modified.

Example: The carry flag is set. The instruction sequence,

```
JNC     LABEL1
CPL     C
JNC     LABEL2
```

will clear the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0 1 0 1	0 0 0 0	rel. address

Operation:

```
JNC
(PC) ← (PC) + 2
IF (C) = 0
   THEN
(PC) ← (PC) + rel
```

JNZ rel

Function: Jump if Accumulator Not Zero

Description: If any bit of the Accumulator is a one, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally holds 00H. The instruction sequence,

```
JNZ     LABEL1
INC     A
JNZ     LABEL2
```

will set the Accumulator to 01H and continue at label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0 1 1 1	0 0 0 0	rel. address

Operation:

```
JNZ
(PC) ← (PC) + 2
IF A ≠ 0
   THEN (PC) ← (PC) + rel
```

JZ rel

Function: Jump if Accumulator Zero

Description: If all bits of the Accumulator are zero, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally holds 01H. The instruction sequence,

```
JZ      LABEL1
DEC     A
JZ      LABEL2
```

will change the Accumulator to 00H and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0 1 1 0	0 0 0 0	rel. address

Operation:

```
JZ
(PC) ← (PC) + 2
IF A = 0
   THEN (PC) ← (PC) + rel
```

LCALL addr16

Function: Long Call

Description: LCALL calls a subroutine located at the indicated address. The instruction adds three to the program counter to generate the address of the next instruction and then pushes the 16-bit result onto the stack (low byte first), incrementing the Stack Pointer by two. The high-order and low-order bytes of the PC are then loaded, respectively, with the second and third bytes of the LCALL instruction. Program execution continues with the instruction at this address. The subroutine may therefore begin anywhere in the full 64k-byte program memory address space. No flags are affected.

Example: Initially the Stack Pointer equals 07H. The label "SUBRTN" is assigned to program memory location 1234H. After executing the instruction,

```
LCALL   SUBRTN
```

at location 0123H, the Stack Pointer will contain 09H, internal RAM locations 08H and 09H will contain 26H and 01H, and the PC will contain 1235H.

Bytes: 3

Cycles: 2

Encoding:

0 0 0 1	0 0 1 0	addr15-addr8	addr7-addr0

Operation:

```
LCALL
(PC) ← (PC) + 3
(SP) ← (SP) + 1
((SP)) ← (PC7-0)
(SP) ← (SP) + 1
((SP)) ← (PC15-8)
(PC) ← addr15-0
```

LJMP addr16 (Implemented in 87C751 and 87C752 for in-circuit emulation only.)

Function: Long Jump

Description: LJMP causes an unconditional branch to the indicated address, by loading the high-order and low-order bytes of the PC (respectively) with the second and third instruction bytes. The destination may therefore be anywhere in the full 64k program memory address space. No flags are affected.

Example: The label "JMPADR" is assigned to the instruction at program memory location 1234H. The instruction,

LJMP JMPADR

at location 0123H will load the program counter with 1234H.

Bytes: 3

Cycles: 2

Encoding:

0 0 0 0	0 0 1 0	addr15-addr8	addr7-addr0

Operation: LJMP
((SP)) ← $addr_{15-8}$

MOV <dest-byte>,<src-byte>

Function: Move byte variable

Description: The byte variable indicated by the second operand is copied into the location specified by the first operand. The source byte is not affected. No other register or flag is affected.

This is by far the most flexible operation. Fifteen combinations of source and destination addressing modes are allowed.

Example: Internal RAM location 30H holds 40H. The value of RAM location 40H is 10H. The data present at input port 1 is 11001010B (0CAH). The instruction sequence,

```
MOV   R0,#30H    ;R0 < = 30H
MOV   A,@R0      ;A < = 40H
MOV   R1,A       ;R1 < = 40H
MOV   B,@R1      ;B < = 10H
MOV   @R1,P1     ;RAM (40H) < = 0CAH
MOV   P2,P1      ;P2 #0CAH
```

leaves the value 30H in register 0, 40H in both the Accumulator and register 1, 10H in register B, and 0CAH (11001010B) both in RAM location 40H and output on port 2.

MOV A,Rn

Bytes: 1

Cycles: 1

Encoding:

1 1 1 0	1 r r r

Operation: MOV
(A) ← (R_n)

***MOV A,direct**

Bytes: 2

Cycles: 1

Encoding:

1 1 1 0	0 1 0 1	direct address

Operation: MOV
(A) ← (direct)

MOV A,@Ri

Bytes: 1

Cycles: 1

Encoding:

1 1 1 0	0 1 1 i

Operation: MOV
(A) ← (R_i)

MOV A,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 1 1	0 1 0 0	immediate data

Operation: MOV
(A) ← #data

MOV Rn,A

Bytes: 1

Cycles: 1

Encoding:

1 1 1 1	1 r r r

Operation: MOV
(R_n) ← (A)

MOV Rn,direct

Bytes: 2

Cycles: 2

Encoding:

1 0 1 0	1 r r r	direct address

Operation: MOV
(R_n) ← (direct)

MOV Rn,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 1 1	1 r r r	immediate data

Operation: MOV
(R_n) ← #data

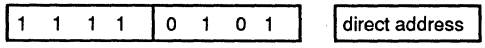
*MOV A,ACC is not a valid instruction.

MOV direct,A

Bytes: 2

Cycles: 1

Encoding:



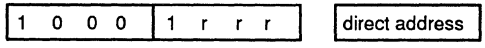
Operation: MOV
(direct) ← (A)

MOV direct,Rn

Bytes: 2

Cycles: 2

Encoding:



Operation: MOV
(direct) ← (R_n)

MOV direct,direct

Bytes: 3

Cycles: 2

Encoding:

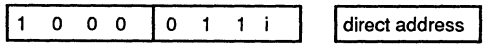
Operation: MOV
(direct) ← (direct)

MOV direct,@Ri

Bytes: 2

Cycles: 2

Encoding:



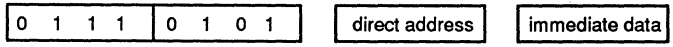
Operation: MOV
(direct) ← (R_i)

MOV direct,#data

Bytes: 3

Cycles: 2

Encoding:



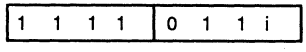
Operation: MOV
(direct) ← #data

MOV @Ri,A

Bytes: 1

Cycles: 1

Encoding:



Operation: MOV
((R_i)) ← (A)

MOV @Ri,direct

Bytes: 2

Cycles: 2

Encoding:

1 0 1 0	0 1 1 i	direct address

Operation: MOV
$((R_i)) \leftarrow$ (direct)

MOV @Ri,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 1 1	0 1 1 i	immediate data

Operation: MOV
$((R_i)) \leftarrow$ #data

MOV <dest-bit>,<src-bit>

Function: Move bit data

Description: The Boolean variable indicated by the second operand is copied into the location specified by the first operand. One of the operands must be the carry flag; the other may be any directly addressable bit. No other register or flag is affected.

Example: The carry flag is originally set. The data present at input Port 3 is 11000101B. The data previously written to output Port 1 is 35H (00110101B). The instruction sequence,

```
MOV   P1.3,C
MOV   C,P3.3
MOV   P1.2,C
```

will leave the carry cleared and change Port 1 to 39H (00111001B).

MOV C,bit

Bytes: 2

Cycles: 1

Encoding:

1 0 1 0	0 0 1 0	bit address

Operation: MOV
(C) $\leftarrow$ (bit)

MOV bit,C

Bytes: 2

Cycles: 2

Encoding:

1 0 0 1	0 0 1 0	bit address

Operation: MOV
(bit) $\leftarrow$ (C

MOV DPTR,#data16

Function: Load Data Pointer with a 16-bit constant

Description: The Data Pointer is loaded with the 16-bit constant indicated. The 16-bit constant is loaded into the second and third bytes of the instruction. The second byte (DPH) is the high-order byte, while the third byte (DPL) holds the low-order byte. No flags are affected.

This is the only instruction which moves 16 bits of data at once.

Example: The instruction,

MOV DPTR,#1234H

will load the value 1234H into the Data Pointer: DPH will hold 12H and DPL will hold 34H.

Bytes: 3

Cycles: 2

Encoding:

1 0 0 1	0 0 0 0	immed. data15-8	immed. data7-0

Operation: MOV

(DPTR) ← (#$data_{15-0}$)

DPH □ DPL ← #$data_{15-8}$ □#$data_{7-0}$

MOVC A,@A+<base-reg>

Function: Move Code byte

Description: The MOVC instructions load the Accumulator with a code byte, or constant from program memory. The address of the byte fetched is the sum of the original unsigned eight-bit Accumulator contents and the contents of a sixteen-bit base register, which may be either the Data Pointer or the PC. In the latter case, the PC is incremented to the address of the following instruction before being added with the Accumulator; otherwise the base register is not altered. Sixteen-bit addition is performed so a carry-out from the low-order eight bits may propagate through higher-order bits. No flags are affected.

Example: A value between 0 and 3 is in the Accumulator. The following instructions will translate the value in the Accumulator to one of four values defined by the DB (define byte) directive:

```
REL_PC:  INC   A
         MOVC  A,@A+PC
         RET
         DB    66H
         DB    77H
         DB    88H
         DB    99H
```

If the subroutine is called with the Accumulator equal to 01H, it will return with 77H in the Accumulator. The INC A before the MOVC instruction is needed to "get around" the RET instruction above the table. If several bytes of code separated the MOVC from the table, the corresponding number would be added to the Accumulator instead.

MOVC A,@A+DPTR

Bytes: 1

Cycles: 2

Encoding:

1 0 0 1	0 0 1 1

Operation: MOVC

(A) ← ((A) + (DPTR))

MOVC A,@A+PC

Bytes: 1

Cycles: 2

Encoding:

1 0 0 0	0 0 1 1

Operation: MOVC
(PC) ← (PC) + 1
(A) ← ((A) + (PC))

MOVX <dest-byte>,<src-byte> (Not implemented in the 8XC752 or 8XC752)

Function: Move External

Description: The MOVX instructions transfer data between the Accumulator and a byte of external data memory, hence the "X" appended to MOV. There are two types of instructions, differing in whether they provide an eight-bit or sixteen-bit indirect address to the external data RAM.

In the first type, the contents of R0 or R1 in the current register bank provide an eight-bit address multiplexed with data on P0. Eight bits are sufficient for external I/O expansion decoding or for a relatively small RAM array. For somewhat larger arrays, port pins can be used to output higher-order address bits. These pins would be controlled by an output instruction preceding the MOVX.

In the second type of MOVX instruction, The Data Pointer generates a sixteen-bit address. P2 outputs the high-order eight address bits (the contents of DPH) while P0 multiplexes the low-order eight bits (DPL) with data. The P2 Special Function Register retains its previous contents while the P2 output buffers are emitting the contents of DPH. This form is faster and more efficient when accessing very large data arrays (up to 64k bytes), since no additional instructions are needed to set up the output ports.

It is possible in some situations to mix the two MOVX types. A large RAM array with its high-order address lines driven by P2 can be addressed via the Data Pointer, or with code to output high-order address bits to P2 followed by a MOVX instruction using R0 or R1.

Example: An external 256 byte RAM using multiplexed address/data lines is connected to the 8051 Port 0. Port 3 provides control lines for the external RAM. Ports 1 and 2 are used for normal I/O. Registers 0 and 1 contain 12H and 34H. Location 34H of the external RAM holds the value 56H. The instruction sequence,

```
MOVX  A,@R1
MOVX  @R0,A
```

copies the value 56H into both the Accumulator and external RAM location 12H.

MOVX A,@Ri

Bytes: 1

Cycles: 2

Encoding:

1 1 1 0	0 0 1 i

Operation: MOVX
(A) ← ((R_i))

MOVX A,@DPTR

Bytes: 1

Cycles: 2

Encoding:

1 1 1 0	0 0 0 0

Operation: MOVX
(A) ← ((DPTR))

MOVX @RI,A

Bytes: 1

Cycles: 2

Encoding:

1	1	1	1	0	0	1	i

Operation: MOVX
$((R_i)) \leftarrow (A)$

MOVX @DPTR,A

Bytes: 1

Cycles: 2

Encoding:

1	1	1	1	0	0	0	0

Operation: MOVX
((DPTR)) ← (A)

MUL AB

Function: Multiply

Description: MUL AB multiplies the unsigned eight-bit integers in the Accumulator and register B. The low-order byte of the sixteen-bit product is left in the Accumulator, and the high-order byte in B. If the product is greater than 255 (0FFH) the overflow flag is set; otherwise it is cleared. The carry flag is always cleared.

Example: Originally the Accumulator holds the value 80 (50H). Register B holds the value 160 (0A0H).The instruction,

MUL AB

will give the product 12,800 (3200H), so B is changed to 32H (00110010B) and the Accumulator is cleared. The overflow flag is set, carry is cleared.

Bytes: 1

Cycles: 4

Encoding:

1	0	1	0	0	1	0	0

Operation: MUL
$(A)_{7\text{-}0} \leftarrow (A) \times (B)$
$(B)_{15\text{-}8}$

NOP

Function: No Operation

Description: Execution continues at the following instruction. Other than the PC, no registers or flags are affected.

Example: It is desired to produce a low-going output pulse on bit 7 of Port 2 lasting exactly 5 cycles. A simple SETB/CLR sequence would generate a one-cycle pulse, so four additional cycles must be inserted. This may be done (assuming are enabled) with the instruction sequence,

```
CLR    P2.7
NOP
NOP
NOP
NOP
SETB   P2.7
```

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	0 0 0 0

Operation: NOP
(PC) ← (PC) + 1

ORL <dest-byte>,<src-byte>

Function: Logical-OR for byte variables

Description: ORL performs the bitwise logical-OR operation between the indicated variables, storing the results in the destination byte. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example: If the Accumulator holds 0C3H (11000011B) and R0 holds 55H (01010101B) then the instruction,

```
ORL   A,R0
```

will leave the Accumulator holding the value 0D7H (11010111B).

When the destination is a directly addressed byte, the instruction can set combinations of bits in any RAM location or hardware register. The pattern of bits to be set is determined by a mask byte, which may be either a constant data value in the instruction or a variable computed in the Accumulator at run-time. The instruction,

```
ORL   P1,#00110010B
```

will set bits 5, 4, and 1 of output Port 1.

ORL A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 1 0 0	1 r r r

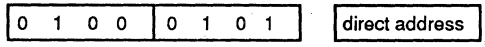
Operation: ORL
(A) ← (A) ∨ (R_n)

ORL A,direct

Bytes: 2

Cycles: 1

Encoding:



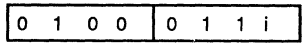
Operation: ORL
(A) ← (A) ∨ (direct)

ORL A,@Ri

Bytes: 1

Cycles: 1

Encoding:



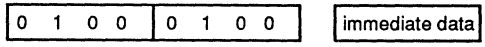
Operation: ORL
(A) ← (A) ∨ ((Ri))

ORL A,#data

Bytes: 2

Cycles: 1

Encoding:



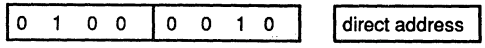
Operation: ORL
(A) ← (A) ∨ #data

ORL direct,A

Bytes: 2

Cycles: 1

Encoding:



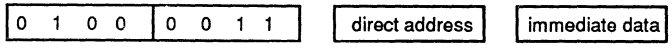
Operation: ORL
(direct) ← (direct) ∨ (A)

ORL direct,#data

Bytes: 3

Cycles: 2

Encoding:



Operation: ORL
(direct) ← (direct) ∨ #data

ORL C,<src-bit>

Function: Logical-OR for bit variables

Description: Set the carry flag if the Boolean value is a logical 1; leave the carry in its current state otherwise. A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected. No other flags are affected.

Example: Set the carry flag if and only if P1.0 = 1, ACC.7 = 1, or OV = 0:

```
ORL   C,P1.0     ;LOAD CARRY WITH INPUT PIN P10
ORL   C,ACC.7    ;OR CARRY WITH THE ACC. BIT 7
ORL   C,/OV      ;OR CARRY WITH THE INVERSE OF OV.
```

ORL C,bit

Bytes: 2

Cycles: 2

Encoding:

0 1 1 1	0 0 1 0	bit address

Operation: ORL

$(C) \leftarrow (C) \vee (bit)$

ORL C,/bit

Bytes: 2

Cycles: 2

Encoding:

1 0 1 0	0 0 0 0	bit address

Operation: ORL

$(C) \leftarrow (C) \vee (\overline{bit})$

POP direct

Function: Pop from stack

Description: The contents of the internal RAM location addressed by the Stack Pointer is read, and the Stack Pointer is decremented by one. The value read is then transferred to the directly addressed byte indicated. No flags are affected.

Example: The Stack Pointer originally contains the value 32H, and internal RAM locations 30H through 32H contain the values 20H, 23H, and 01H, respectively. The instruction sequence,

```
POP   DPH
POP   DPL
```

will leave the Stack Pointer equal to the value 30H and the Data Pointer set to 0123H. At this point the instruction,

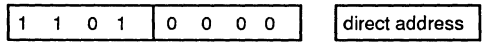
```
POP   SP
```

will leave the Stack Pointer set to 20H. Note that in this special case the Stack Pointer was decremented to 2FH before being loaded with the value popped (20H).

Bytes: 2

Cycles: 2

Encoding:



Operation:

```
POP
(direct) ← ((SP))
(SP) ← (SP) – 1
```

PUSH direct

Function: Push onto stack

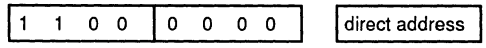
Description: The Stack Pointer is incremented by one. The contents of the indicated variable is then copied into the internal RAM location addressed by the Stack Pointer. Otherwise no flags are affected.

Example: On entering an interrupt routine the Stack Pointer contains 09H. The Data Pointer holds the value 0123H. The instruction sequence,

```
PUSH  DPL
PUSH  DPH
```

will leave the Stack Pointer set to 0BH and store 23H and 01H in internal RAM locations 0AH and 0BH, respectively.

Encoding:



Operation:

```
PUSH
(SP) ← (SP) + 1
((SP)) ← (direct)
```

RET

Function: Return from subroutine

Description: RET pops the high- and low-order bytes of the PC successively from the stack, decrementing the Stack Pointer by two. Program execution continues at the resulting address, generally the instruction immediately following an ACALL or LCALL. No flags are affected.

Example: The Stack Pointer originally contains the value 0BH. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The instruction,

RET

will leave the Stack Pointer equal to the value 09H. Program execution will continue at location 0123H.

Bytes: 1

Cycles: 2

Encoding:

0	0	1	0	0	0	1	0

Operation:

RET
$(PC_{15-8}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$
$(PC_{7-0}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$

RETI

Function: Return from interrupt

Description: RETI pops the high- and low-order bytes of the PC successively from the stack, and restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed. The Stack Pointer is left decremented by two. No other registers are affected; the PSW is not automatically restored to its pre-interrupt status. Program execution continues at the resulting address, which is generally the instruction immediately after the point at which the interrupt request was detected. If a lower- or same-level interrupt has been pending when the RETI instruction is executed, that one instruction will be executed before the pending interrupt is processed.

Example: The Stack Pointer originally contains the value 0BH. An interrupt was detected during the instruction ending at location 0122H. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The instruction,

RETI

will leave the Stack Pointer equal to 09H and return program execution to location 0123H.

Bytes: 1

Cycles: 2

Encoding:

0	0	1	1	0	0	1	0

Operation:

RETI
$(PC_{15-8}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$
$(PC_{7-0}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$

RL A

Function: Rotate Accumulator Left

Description: The eight bits in the Accumulator are rotated one bit to the left. Bit 7 is rotated into the bit 0 position. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,

RL A

leaves the Accumulator holding the value 8BH (10001011B) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0	0	1	0	0	0	1	1

Operation:

RL
$(A_{n+1}) \leftarrow (A_n)$, n = 0 – 6
(A0) ← (A7)

RLC A

Function: Rotate Accumulator Left through the Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the left. Bit 7 moves into the carry flag; the original state of the carry flag moves into the bit 0 position. No other flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B), and the carry is zero. The instruction,

RLC A

leaves the Accumulator holding the value 8BH (10001010B) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	0	0	1	1

Operation:

RLC
$(A_{n+1}) \leftarrow (A_n)$, n = 0 – 6
(A0) ← (C)
(C) ← (A7)

RR A

Function: Rotate Accumulator Right

Description: The eight bits in the Accumulator are rotated one bit to the right. Bit 0 is rotated into the bit 7 position. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,

RR A

leaves the Accumulator holding the value 0E2H (11100010B) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	0	0	1	1

Operation: RR

$(A_n) \leftarrow (A_{n+1})$, n = 0 – 6

(A7) ← (A0)

RRC A

Function: Rotate Accumulator Right through the Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the right. Bit 0 moves into the carry flag; the original state of the carry flag moves into the bit 7 position. No other flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B), and the carry is zero. The instruction,

RRC A

leaves the Accumulator holding the value 62 (01100010B) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1	0	0	1	1

Operation: RRC

$(A_n) \leftarrow (A_{n+1})$, n = 0 – 6

(A7) ← (C)

(C) ← (A0)

SETB <bit>

Function: Set Bit

Description: SETB sets the indicated bit to one. SETB can operate on the carry flag or any directly addressable bit. No other flags are affected.

Example: The carry flag is cleared. Output Port 1 has been written with the value 34H (00110100B). The instructions,

```
SETB  C
SETB  P1.0
```

will leave the carry flag set to 1 and change the data output on Port 1 to 35H (00110101B).

SETB C

Bytes: 1

Cycles: 1

Encoding:

1 1 0 1	0 0 1 1

Operation:
SETB
(C) ← 1

SETB bit

Bytes: 2

Cycles: 1

Encoding:

1 1 0 1	0 0 1 0	bit address

Operation:
SETB
(bit) ← 1

SJMP rel

Function: Short Jump

Description: Program control branches unconditionally to the address indicated. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction to 127 bytes following it.

Example: The label "RELADR" is assigned to an instruction at program memory location 0123H. The instruction,

```
SJMP  RELADR
```

will assemble into location 0100H. After the instruction is executed, the PC will contain the value 0123H.

(*Note:* Under the above conditions the instruction following SJMP will be at 102H. Therefore, the displacement byte of the instruction will be the relative offset (0123H-0102H) = 21H. Put another way, an SJMP with a displacement of 0FEH would be a one-instruction infinite loop.)

Bytes: 2

Cycles: 2

Encoding:

1 0 0 0	0 0 0 0	rel. address

Operation:
SJMP
(PC) ← (PC) + 2
(PC) ← (PC) + rel

SUBB A, <src-byte>

Function: Subtract with borrow

Description: SUBB subtracts the indicated variable and the carry flag together from the Accumulator, leaving the result in the Accumulator. SUBB sets the carry (borrow) flag if a borrow is needed for bit 7, and clears C otherwise. (If C was set *before* executing a SUBB instruction, this indicates that a borrow was needed for the previous step in a multiple precision subtraction, so the carry is subtracted from the Accumulator along with the source operand.) AC is set if a borrow is needed for bit 3, and cleared otherwise. OV is set if a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not bit 6.

When subtracting signed integers OV indicates a negative number produced when a negative value is subtracted from a positive value, or a positive result when a positive number is subtracted from a negative number.

The source operand allows four addressing modes: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C9H (11001001B), register 2 holds 54H (01010100B), and the carry flag is set. The instruction,

SUBB A,R2

will leave the value 74H (01110100B) in the Accumulator, with the carry flag and AC cleared but OV set.

Notice that 0C9H minus 54H is 75H The difference between this and the above result is due to the carry (borrow) flag being set before the operation. If the state of the carry is not known before starting a single or multiple-precision subtraction, it should be explicitly cleared by a CLR C instruction

SUBB A,Rn

Bytes: 1

Cycles: 1

Encoding:

1	0	0	1	1	r	r	r

Operation: SUBB
(A) ← (A) – (C) – (R_n)

SUBB A,direct

Bytes: 2

Cycles: 1

Encoding:

1	0	0	1	0	1	0	1	direct address

Operation: SUBB
(A) ← (A) – (C) – (direct)

SUBB A,@Ri

Bytes: 1

Cycles: 1

Encoding:

1	0	0	1	0	1	1	i

Operation: SUBB
(A) ← (A) – (C) – (R_i)

SUBB A,#data

Bytes: 1

Cycles: 1

Encoding:

1	0	0	1	0	1	0	0	immediate data

Operation: SUBB
(A) ← (A) – (C) – (#data)

SWAP A

Function: Swap nibbles within the Accumulator

Description: SWAP A interchanges the low- and high-order nibbles (four-bit fields) of the Accumulator (bits 3-0 and bits 7-4). The operation can also be thought of as a four-bit rotate instruction. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,

SWAP A

leaves the Accumulator holding the value 5CH (01011100B).

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0	0 1 0 0

Operation: SWAP
$(A_{3-0}) \rightleftarrows (A_{7-4})$

XCH A,<byte>

Function: Exchange Accumulator with byte variable

Description: XCH loads the Accumulator with the contents of the indicated variable, at the same time writing the original Accumulator contents to the indicated variable. The source/destination operand can use register, direct, or register-indirect addressing.

Example: R0 contains the address 20H. The Accumulator holds the value 3FH (00111111B). Internal RAM location 20H holds the value 75H (01110101B). The instruction,

XCH A,@R0

will leave the RAM location 20H holding the values 3FH (00111111B) and 75H (01110101B) in the Accumulator.

XCH A,Rn

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0	1 r r r

Operation: XCH
$(A) \rightleftarrows (R_n)$

XCH A,direct

Bytes: 2

Cycles: 1

Encoding:

1 1 0 0	0 1 0 1	direct address

Operation: XCH
$(A) \rightleftarrows$ (direct)

XCH A,@Ri

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0	0 1 1 i

Operation: XCH
$(A) \rightleftarrows ((R_i))$

XCHD A,@Ri

Function: Exchange Digit

Description: XCHD exchanges the low-order nibble of the Accumulator (bits 3-0), generally representing a hexadecimal or BCD digit, with that of the internal RAM location indirectly addressed by the specified register. The high-order nibbles (bits 7-4) of each register are not affected. No flags are affected.

Example: R0 contains the address 20H. The Accumulator holds the value 36H (00110110B). Internal RAM location 20H holds the value 75H (01110101B). The instruction,

XCHD A,@R0

will leave RAM location 20H holding the value 76H (01110110B) and 35H (00110101B) in the Accumulator.

Bytes: 1

Cycles: 1

Encoding:

1 1 0 1	0 1 1 i

Operation: XCHD
$(A_{3\text{-}0}) \rightleftarrows ((Ri_{3\text{-}0}))$

XRL <dest-byte>,<src-byte>

Function: Logical Exclusive-OR for byte variables

Description: XRL performs the bitwise logical Exclusive-OR operation between the indicated variables, storing the results in the destination. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

(*Note:* When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.)

Example: If the Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) then the instruction,

XRL A,R0

will leave the Accumulator holding the value 69H (01101001B).

When the destination is a directly addressed byte, this instruction can complement combinations of bits in any RAM location or hardware register. The pattern of bits to be complemented is then determined by a mask byte, either a constant contained in the instruction or a variable computed in the Accumulator at run-time. The instruction,

XRL P1,#00110001B

will complement bits 5, 4, and 0 of output Port 1.

XRL A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 1 1 0	1 r r r

Operation: XRL
(A) ← (A) ⩛ (R_n)

XRL A,direct

Bytes: 2

Cycles: 1

Encoding:

0 1 1 0	0 1 0 1	direct address

Operation: XRL
(A) ← (A) ⩛ (direct)

XRL A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 1 1 0	0 1 1 i

Operation: XRL
(A) ← (A) ⩛ (R_i)

XRL A,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 1 0	0 1 0 0	immediate data

Operation: XRL
(A) ← (A) ⩛ #data

XRL direct,A

Bytes: 2

Cycles: 1

Encoding:

0 1 1 0	0 0 1 0	direct address

Operation: XRL
(direct) ← (direct) ⩛ (A)

XRL direct,#data

Bytes: 3

Cycles: 2

Encoding:

0 1 1 0	0 0 1 1	direct address	immediate data

Operation: XRL
(direct) ← (direct) ⩛ #data

EPROM PRODUCTS

Most of the 80C51 derivative products offered by Philips are supported with an EPROM version. Currently available EPROM parts are the 87C51, 87C451, 87C552, 87C52, 87C752, and the 87C751. EPROM versions of the 87C550, 87C652, 87C654, and 87C528 are now in development.

All EPROM products are available in both windowed DIP and OTP package configurations. The windowed DIP package allows the EPROM to be erased under a strong UV light source, making program development easier and faster. The OTP (One Time Programmable) version cannot be erased because there is no window through which the die could be exposed to UV light. While the EPROM can only be programmed once in the OTP package, the part costs less than in windowed DIP and therefore offers an advantage for those not desiring to use the masked ROM version of the part.

The EPROM products are fully supported on the industry standard EPROM programmers.

Programming the 87C51

The setup for programming the microcontroller is shown in Figure 1. Note that the part is running with a 4 to 6 MHz oscillator. The clock must be running because the device is executing internal address and program data transfers during the programming.

To program the 87C51, the address of the EPROM location to be programmed is applied to ports 1 and 2 as shown in Figure 1. The code byte to be programmed into this location is applied to port 0. RST, $\overline{PSEN}$, and the pins of ports 2 and 3 specified in Table 1 are held at the "Program Code Data" levels specified in the table. The ALE/$\overline{PROG}$ is then pulsed low 25 times to program the addressed location.

Encryption Table

The encryption table is a feature of the 87C51, and its derivatives, that protects the code from being easily read by anyone other than the programmer. The encryption table is 16 to 64 bytes of code, depending on the microcontroller, that are exclusive NORed with the program code data as it is read out. The first byte is XNORed with the first location read, the second with the second read, etc. through the sixteenth byte read. The seventeenth byte is XNORed with the first byte of the encryption table, the eighteenth with the second, etc. and on in sixteen-byte groups.

After the Encryption table has been programmed the user has to know its contents in order to correctly decode the program code data. The encryption table itself cannot be read out.

The encryption table is programmed in the same manner as the program memory, but using the "Pgm Encryption Table" levels specified in Table 1. After the encryption table is programmed, verification cycles will produce only encrypted information.

Lock Bit

There are two lock bits on the 87C51 that, when set, prevent the program data memory from being read out or programmed further. To program the lock bits, repeat the programming sequence using the "Pgm Lock Bit" levels specified in Table 1.

After the first lock bit is programmed, further programming of the code memory or the encryption table is disabled. The other lock bit can of course still be programmed. With only lock bit one programmed, the memory can still be read out for program verification. After the second lock bit is programmed, it is no longer possible to read out (verify) the program memory.

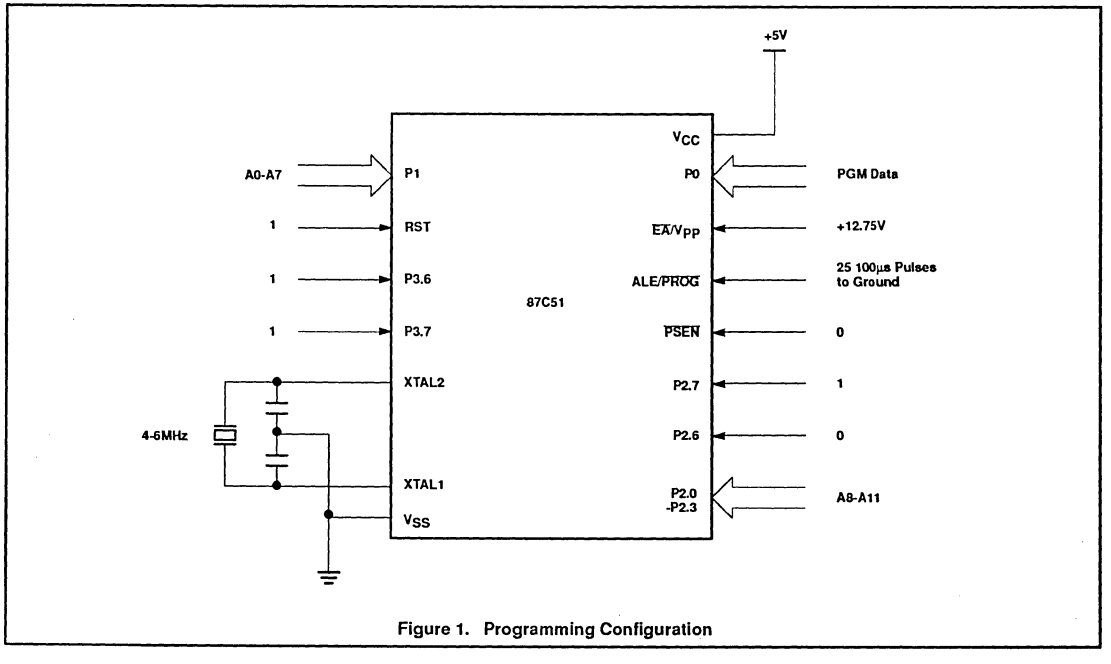


Figure 1. Programming Configuration

Table 1. EPROM Programming Modes

MODE	RST	PSEN	ALE/PROG	EA/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:
1. "0" = valid low for that pin, "1" = valid high for that pin.
2. V_{PP} = 12.75 ±0.25V.
3. V_{CC} = 5V ±10% during programming and verification.

* ALE/PROG receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100ms (±10µs) and high for a minimum of 10µs.

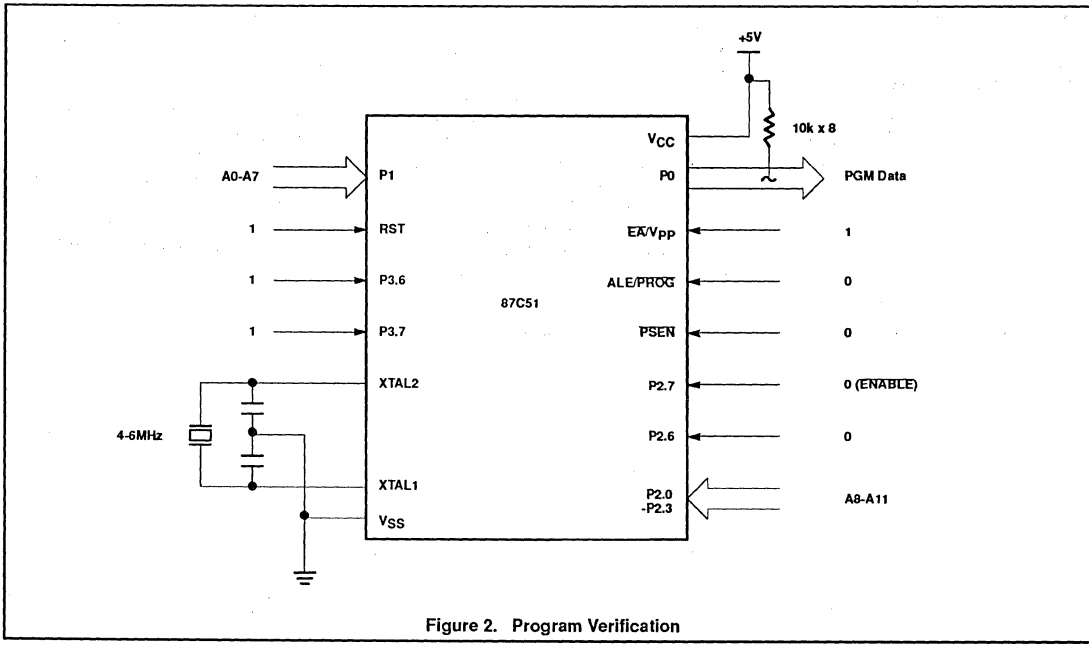


Figure 2. Program Verification

Program Verification

If lock bit 2 has not been programmed the on-chip program memory can be read out for program verification. To verify the contents of the program memory, the address of the location to be read is applied to ports 1 and 2 as shown in Figure 2. The other pins are held at the "Verify Code Data" levels indicated in Table 1. The contents of the addressed location will appear on port 0. For this operation external pull-ups are required on port 0 as shown in Figure 2. Note that if the encryption table has been programmed the data presented at port 0 will be the exclusive NOR of the program byte with a byte from the encryption table.

Signature Bytes

The 87C51 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C51 manufactured by Philips.

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates the part is made by Philips

(031H) =

90H 87C451	97H 87C52
92H 87C51	99H 87C654
93H 87C652	9AH 80C52
94H 87C552	9BH 87C528
96H 87C550	9CH 87C592

EPROM Erasure

Erasure of the EPROM occurs when the chip is exposed to light with wavelengths shorter than 4000 angstroms. Sunlight and fluorescent lighting have wavelengths in this range, so exposure to these light sources over an extended period of time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. It is recommended, for this reason, that an opaque label be placed over the window. If the part is subject to elevated temperatures or an environment where solvents are used, Kapton tape (Fluorglas part number 2345-5 or its equivalent) can be used.

The recommended erasure procedure is to expose the chip to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000µW/cm^2 rating for 20 to 40 minutes, at a distance of 1 inch, is adequate.

Programming the 87C751 and 87C752

The 87C751 and 87C752 are programmed using a Quick-pulse programming algorithm that is similar to that used for the 87C51. It differs from the 87C51 in that a serial data stream is used to place the 87C751 in the programming mode.

Figure 3 shows a block diagram of the programming configuration for the 87C751. Port pin P0.2 is used for the programming voltage supply input (V_{PP} signal). Port pin P0.1 is used for the program (PGM) signal.

Port 3 accepts the address input for the EPROM location to be programmed. Both the high and low components of the eleven-bit address are presented to the part through port 3. Multiplexing of the address components is performed using ASEL (P0.0).

Port 1 is used as a bidirectional data bus during programming and verify operations. During the programming mode, it accepts the byte to be programmed. In the verify mode, it returns the contents of the specified address location.

The X1 pin is the oscillator input and receives the master system clock. This clock should be between 1.2 and 6MHz.

The RESET pin is used to accept the serial data stream that places the 87C751 into various programming modes. This pattern consists of a 10-bit code with the LSB send first. Each bit is synchronized to the clock input X1.

To program the 87C751 the part must be put into the programming mode by presenting the proper serial code (see Table 2) to the RESET pin. To do this RESET should be held high for at least two machine cycles. Port pins P0.1 and P0.2 will be at VOH as a result of this, but they must be driven high prior to sending the serial data stream on the RESET pin. The serial data bits can now be transmitted over the RESET pin placing the 87C751 into one of the programming modes. Following the transmission of the last data bit, the reset pin should be held low.

Next the address information for the location to be programmed is placed on Port 3 and ASEL is used to perform the address multiplexing. ASEL should be driven high and then Port 3 driven with the high-order address bits. ASEL is then driven low, latching the high-order bits internally. Port 3 can now be driven with the low 8 bits of the address, completing the addressing of the location to be programmed.

A high-voltage V_{PP} level is now applied to the V_{PP} input. This sets Port 1 as an input port. The data to be programmed to the EPROM array should be placed on Port 1. A series of 25 programming pulses is now applied to the PGM pin (P0.1) to program the addressed EPROM location.

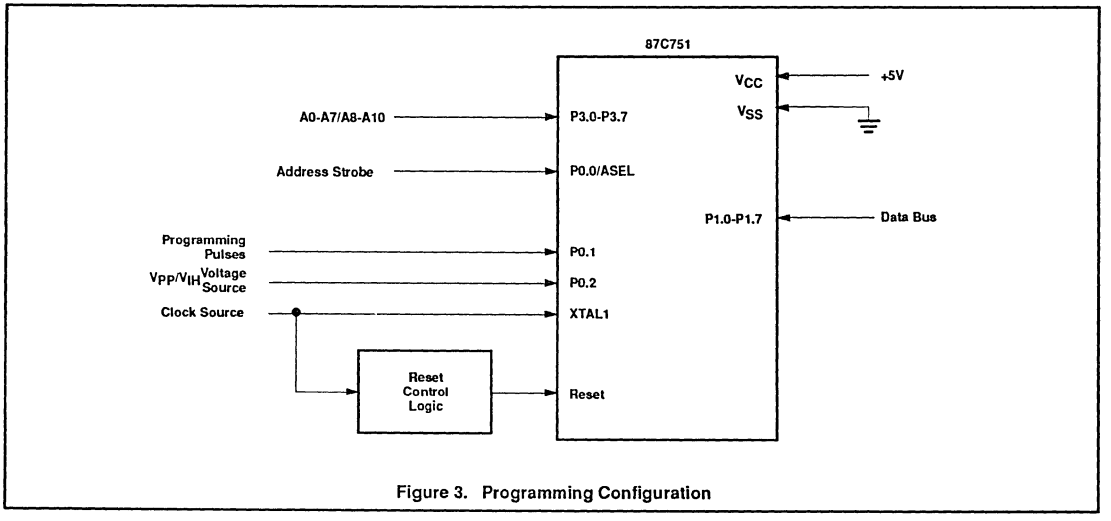


Figure 3. Programming Configuration

Table 2. Serial Codes

OPERATION	SERIAL CODE	P0.1 (PGM/)	P0.2 (V_{PP})
Program user EPROM	296H	–*	V_{PP}
Verify user EPROM	296H	V_{IH}	V_{IH}
Program key EPROM	292H	–*	V_{PP}
Verify key EPROM	292H	V_{IH}	V_{IH}
Program security bit 1	29AH	–*	V_{PP}
Program security bit 2	298H	–*	V_{PP}
Verify security bits	29AH	V_{IH}	V_{IH}
Read signature bytes	19CH	V_{IH}	V_{IH}

NOTE:
* Pulsed from V_{IH} to V_{IL} and returned to V_{IH}.

Program Verification

The EPROM array can be verified by placing the part in the programming mode as described above and forcing the V_{PP} pin to the V_{OH} level. Four machine cycles after addressing a location the contents of the addressed location will appear on Port 1.

87C751 and 87C752 Signature Bytes

The signature bytes for the 87C751 and 87C752 are read differently and are in different locations than those on the 87C51. Due to its reduced pin count, the part has to be put into "Signature Byte Read Mode" by placing a 10-bit serial data stream on the Reset pin. The proper code and the conditions of P0.1 and P0.2, for this mode, are shown in Table 2.

Once the part has been placed into the Signature Byte Read Mode, the signature bytes can be read by the same procedure as a normal verification of locations 01EH and 01FH. The values are:

01EH = 15H indicates the part is made by Philips
01FH = 91H - 87C751
01FH = 95H - 87C752

Programming Features

The 87C751 has all of the special programming features incorporated within its EPROM array that the 87C51 has. It has an encryption key table and two security bits (lock bits). These function exactly as they do in the 87C51. They are programmed or verified by sending the proper code over the RESET pin (see Table 2) and then following the 87C751 programming procedure as described previously.

Erasure Characteristics

The erasure procedure is exactly the same as that described for the 87C51.

Single-chip 8-bit microcontroller — 8031AH/8051AH

DESCRIPTION

The Philips 8031AH/8051AH is a high-performance microcontroller fabricated with Philips high-density highly reliable +5V, depletion-load, N-channel, silicon-gate, N500 MOS process technology. It provides the hardware features, architectural enhancements and instructions that are necessary to make it a powerful and cost-effective controller for applications requiring up to 64k bytes of program memory and/or up to 64k bytes of data storage.

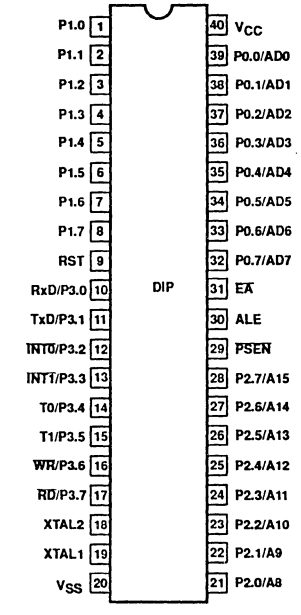
The 8051AH contains a 4k × 8 read-only program memory, a 128 × 8 read-only data memory, 32 I/O lines, two 16-bit counter/timers, a five-source, two-priority level nested interrupt structure, a serial I/O port for either multi-processor communications, I/O expansion or full duplex UART, and on-chip oscillator and clock circuits. The 8031AH is identical, except that it lacks the program memory. For systems that require extra capability, the 8051AH can be expanded using standard TTL compatible memories and byte oriented peripheral controllers.

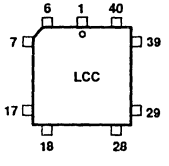
The 8051AH microcontroller, like its 8048 predecessor, is efficient both as a controller and as an arithmetic processor. It has extensive facilities for binary and BCD arithmetic and excels in bit-handling capabilities. Efficient use of program memory results from an instruction set consisting of 44% one-byte, 41% two-byte, and 15% three-byte instructions. With a 12MHz crystal, 58% of the instructions execute in 1μs, 40% in 2μs and multiply and divide require only 4μs.

FEATURES

- Reduced supply current
- 4k × 8 ROM (8051AH)
- 128 × 8 RAM
- Four 8-bit ports, 32 I/O lines
- Two 16-bit timer/event counters
- High-performance full-duplex serial channel
- External memory expandable to 128k
- Boolean processor
- Industry standard 8051 architecture:
  - Non-paged jumps
  - Direct addressing
  - Four 8-register banks
  - Stack depth up to 128-bytes
  - Multiply, divide, subtract, compare
- Most instructions execute in 1μs
- 4μs multiply and divide

PIN CONFIGURATIONS



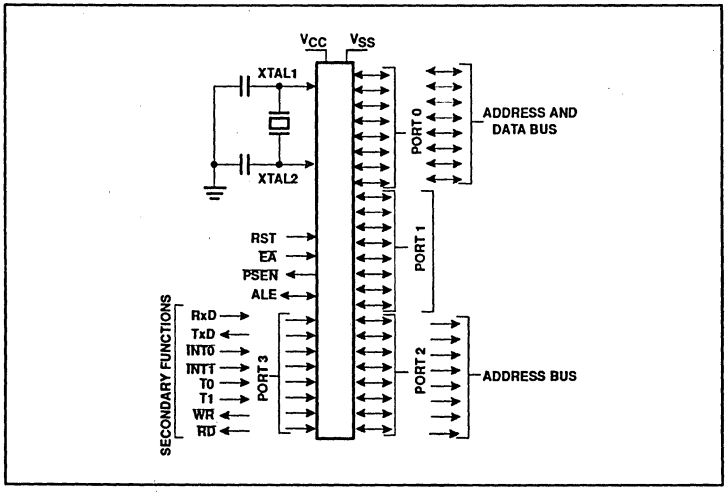


Pin	Function	Pin	Function	Pin	Function
1	NC	15	INT1/P3.3	30	P2.6/A14
2	P1.0	16	T0/P3.4	31	P2.7/A15
3	P1.1	17	T1/P3.5	32	PSEN
4	P1.2	18	WR/P3.6	33	ALE
5	P1.3	19	RD/P3.7	34	NC
6	P1.4	20	XTAL2	35	EA
7	P1.5	21	XTAL1	36	P0.7/AD7
8	P1.6	22	V_{SS}	37	P0.6/AD6
9	P1.7	23	NC	38	P0.5/AD5
10	RST	24	P2.0/A8	39	P0.4/AD4
11	RxD/P3.0	25	P2.1/A9	40	P0.3/AD3
12	NC	26	P2.2/A10	4'1	P0.2/AD2
13	TxD/P3.1	27	P2.3/A11	42	P0.1/AD1
14	INT0/P3.2	28	P2.4/A12	43	P0.0/AD0
		29	P2.5/A13	44	V_{CC}

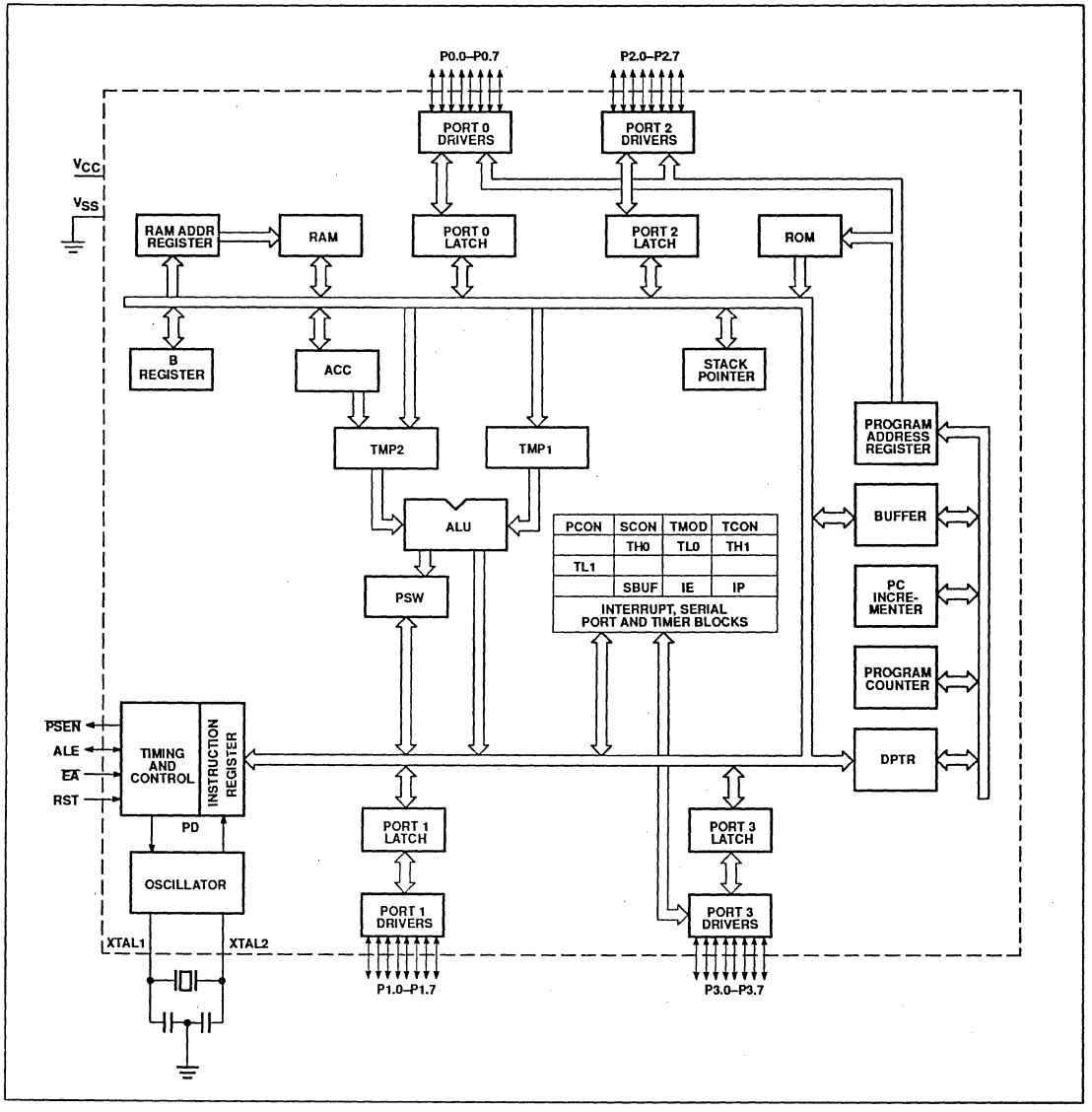
PART NUMBER SELECTION

PHILIPS		SIGNETICS		TEMPERATURE °C AND PACKAGE	FREQUENCY MHz
ROMless	ROM	ROMless	ROM		
MAF8031AH2-12P	MAF8051AH-2P	SCN8031HACN40	SCN8051HACN40	–40 to +85, plastic DIP	12
MAB8031AH2-12P	MAB8051AH-2P	SCN8031HCCN40	SCN8051HCCN40	0 to +70, plastic DIP	12
		SCN8031HCFN40	SCN8051HCFN40	0 to +70, plastic DIP	15
		SCN8031HAFN40	SCN8051HAFN40	–40 to +85, plastic DIP	15
MAB8031AH2-12WP	MAB8051AH-2WP	SCN8031HCCA44	SCN8051HCCA44	0 to +70, plastic LCC	12
MAF8031AH2-12WP	MAF8051AH-2WP	SCN8031HACA44	SCN8051HACA44	–40 to +85, plastic LCC	12
		SCN8031HCFA44	SCN8051HCFA44	0 to +70, plastic LCC	15
		SCN8031HAFA44	SCN8051HAFA44	–40 to +85, plastic LCC	15

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTIONS

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	20	22	I	**Ground:** 0V reference.
V_{CC}	40	44	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–36	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s.
P1.0–P1.7	1–8	2–9	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}).
P2.0–P2.7	21–28	24–31	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 80C51 family, as listed below:
	10	11	I	**RxD (P3.0):** Serial input port
	11	13	O	**TxD (P3.1):** Serial output port
	12	14	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	I	**T0 (P3.4):** Timer 0 external input
	15	17	I	**T1 (P3.5):** Timer 1 external input
	16	18	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE	30	33	I/O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory.
$\overline{PSEN}$	29	32	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$	31	35	I	**External Access Enable:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 0FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 0FFFH.
XTAL1	19	21	I	**Crystal 1:** Input to the inverting oscillator amplifier.
XTAL2	18	20	O	**Crystal 2:** Output from the inverting oscillator amplifier and input to the internal clock generator circuits.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

DESIGN CONSIDERATIONS

At power-on, the voltage on V_{CC} and RST should come up at the same time for a proper start-up.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage on any other pin to V_{SS}	–0.5 to +7.0	V
Input, output current on any single pin	10	mA
Power dissipation	1.0	W

DC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C, V_{CC} = 5V ±10%, V_{SS} = 0V[4, 5]

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNIT
			MIN	MAX	
V_{IL}	Input low voltage		–0.5	0.8	V
V_{IH}	Input high voltage; except XTAL2, RST		2.0	V_{CC}+0.5	V
V_{IH1}	Input high voltage to RST for reset, XTAL2	XTAL1 to V_{SS}	2.5	V_{CC}+0.5	V
V_{OL}	Output low voltage; ports 1, 2, 3[6]	I_{OL} = 1.6mA		0.45	V
V_{OL1}	Output low voltage; port 0, ALE, $\overline{PSEN}$[6]	I_{OL} = 3.2mA		0.45	V
V_{OH}	Output high voltage; ports 1, 2, 3	I_{OH} = –80uA	2.4		V
V_{OH1}	Output high voltage; port 0, ALE, $\overline{PSEN}$[3]	I_{OH} = –400uA	2.4		V
I_{IL}	Logical 0 input current; ports 1, 2, 3	V_{IN} = 0.45V		–500	µA
I_{IH1}	Input high current to RST for reset	V_{IN} < V_{CC} –1.5V		500	µA
I_{LI}	Input leakage current; port 0, $\overline{EA}$	0.45 < V_{IN} < V_{CC}		±10	µA
I_{IL2}	Logical 0 input current for XTAL2	XTAL1 = V_{SS}, V_{IN} = 0.45V		–3.2	mA
I_{CC}	Power supply current	All outputs disconnected and $\overline{EA}$ = V_{CC}		125	mA
C_{IO}	Pin capacitance			10	pF

T_{amb} = –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNIT
			MIN	MAX	
V_{IH}	Input high voltage; except XTAL2, RST		2.1	V_{CC}+0.5	V
V_{IH1}	Input high voltage to RST and XTAL2	XTAL1 = V_{SS}	2.6	V_{CC}+0.5	V
I_{IL2}	Logical 0 input current for XTAL2	XTAL1 = V_{SS}, V_{IN} = 0.45V		–4.0	mA
I_{CC}	Power supply current	All outputs disconnected and $\overline{EA}$ = V_{CC}		135	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.
5. All voltage measurements are referenced to ground. For testing, all input signals swing between 0.45V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and at output voltages of 0.8V and 2.0V as appropriate.
6. V_{OL} is derated when the device rapidly discharges external capacitance. This AC noise is most pronounced during emission of address data. When using external memory, locate the latch or buffer as close as possible to the device.

Datum	Emitting Ports	Degraded I/O Lines	V_{OL} (Peak Max)
Address	P2, P0	P1, P3	0.8V
Write Data	P0	P1, p3, ALE	0.8V

7. C_L = 100pF for port 0, ALE and $\overline{PSEN}$ outputs: C_L = 80pF for all other ports.

AC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or −40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V[1,2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$		Oscillator frequency: **Speed Versions** SCN8051/31 C MAB8051/31 −2 MAF8051/31 −2 SCN8051/31 F			 3.5 3.5 3.5 3.5	 12 12 12 15	 MHz MHz MHz MHz
t_{LHLL}	1	ALE pulse width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	43		$t_{CLCL}-40$		ns
t_{LLAX}	1	Address hold after ALE low	48		$t_{CLCL}-35$		ns
t_{LLIV}	1	ALE low to valid instruction in		233		$4t_{CLCL}-100$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	58		$t_{CLCL}-25$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	215		$3t_{CLCL}-35$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		125		$3t_{CLCL}-125$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		63		$t_{CLCL}-20$	ns
t_{AVIV}	1	Address to valid instruction in		302		$5t_{CLCL}-115$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		20		20	ns
t_{PXAV}	1	$\overline{PSEN}$ to address valid	75		$t_{CLCL}-8$		ns
Data Memory							
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}-165$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		97		$2t_{CLCL}-70$	ns
t_{LLDV}	2, 3	ALE low to valid data in		517		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		585		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}-130$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	23		$t_{CLCL}-60$		ns
t_{QVWH}	2, 3	Data valid to $\overline{WR}$ high	433		$7t_{CLCL}-150$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	33		$t_{CLCL}-50$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		20		20	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	5	High time	20		20		ns
t_{CLCX}	5	Low time	20		20		ns
t_{CLCH}	5	Rise time		20		20	ns
t_{CHCL}	5	Fall time		20		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{\text{PSEN}}$
Q – Output data
R – $\overline{\text{RD}}$ signal
t – Time
V – Valid
W – $\overline{\text{WR}}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{\text{PSEN}}$ low.

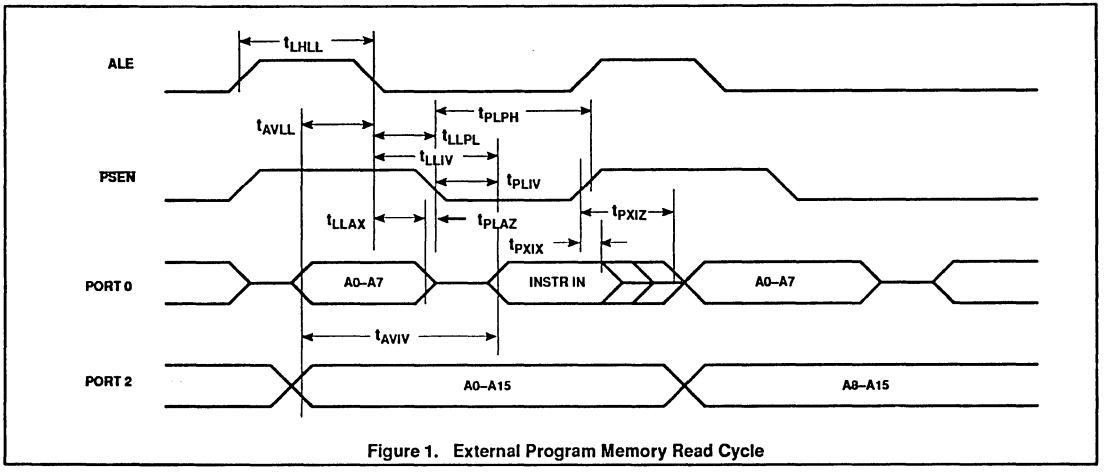


Figure 1. External Program Memory Read Cycle

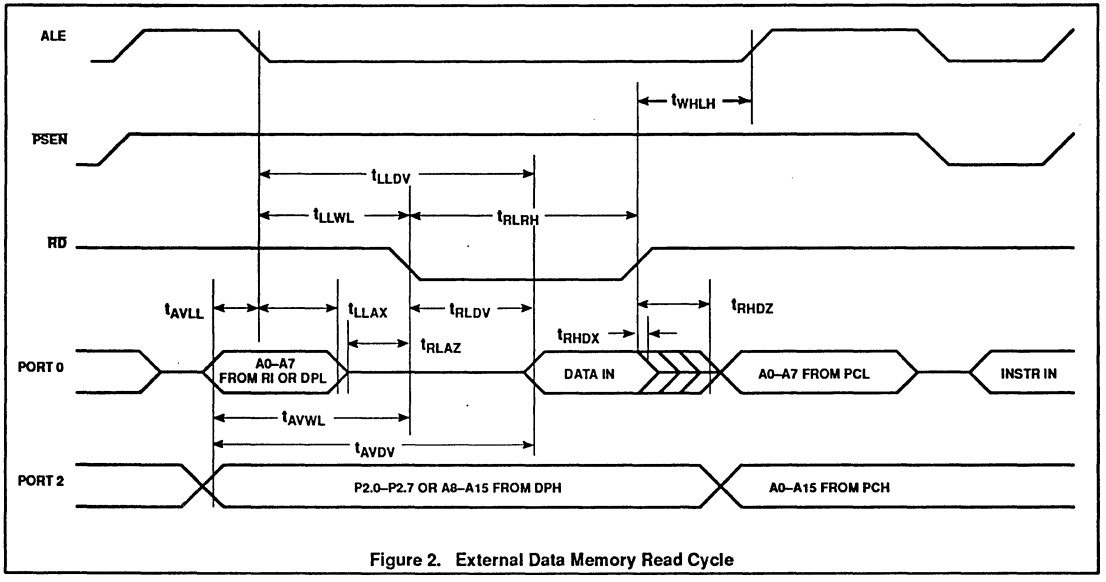


Figure 2. External Data Memory Read Cycle

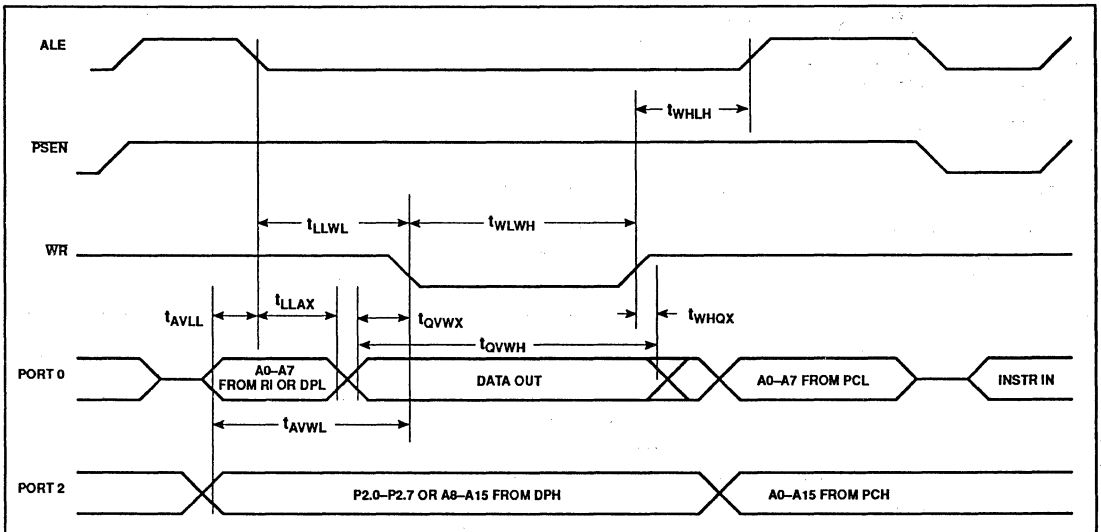


Figure 3. External Data Memory Write Cycle

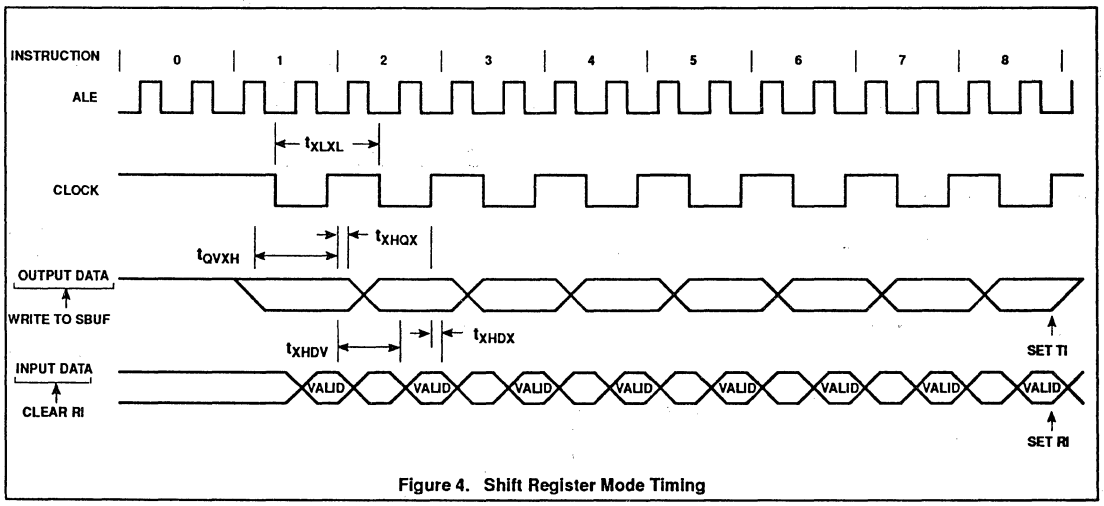


Figure 4. Shift Register Mode Timing

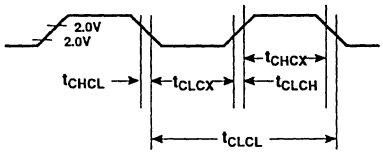
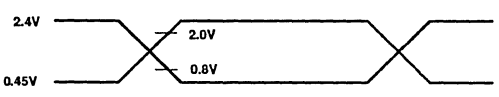
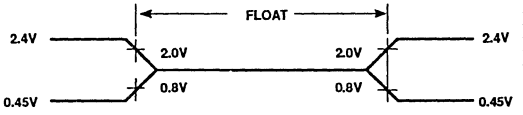


Figure 5. External Clock Drive



NOTE:
AC inputs during testing are driven at 2.4V for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at 2.0V min for a logic '1' and 0.8V for a logic '0'.

Figure 6. AC Testing Input/Output



NOTE:
For timing purposes, the float state is defined as the point at which a P0 pin sinks 2.4mA or sources 400µA at the voltage test levels.

Figure 7. Float Waveform

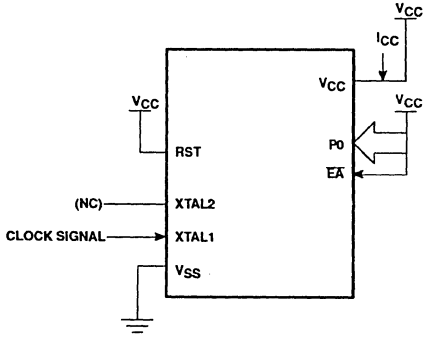


Figure 8. I_{CC} Test Condition, Active Mode
All other pins are disconnected

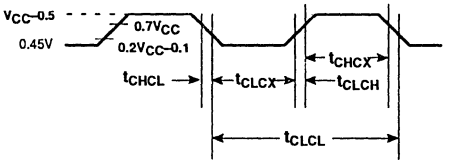


Figure 9. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

CMOS single-chip 8-bit microcontroller — 80C31/80C51/87C51

DESCRIPTION

The Philips 80C31/80C51/87C51 is a high-performance microcontroller fabricated with Philips high-density CMOS technology. The CMOS 8XC51 is functionally compatible with the NMOS 8031/8051 microcontrollers. The Philips CMOS technology combines the high speed and density characteristics of HMOS with the low power attributes of CMOS. Philips epitaxial substrate minimizes latch-up sensitivity.

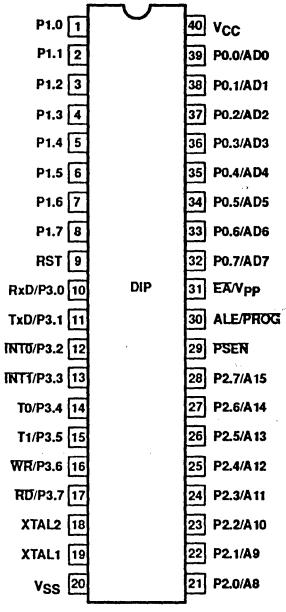
The 8XC51 contains a 4k × 8 ROM (80C51) EPROM (87C51), a 128 × 8 RAM, 32 I/O lines, two 16-bit counter/timers, a five-source, two-priority level nested interrupt structure, a serial I/O port for either multi-processor communications, I/O expansion or full duplex UART, and on-chip oscillator and clock circuits.

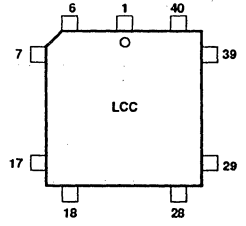
In addition, the device has two software selectable modes of power reduction — idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

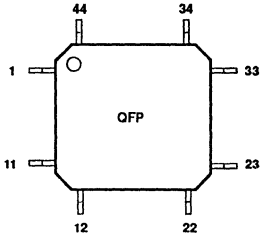
FEATURES

- 8031/8051 compatible
  - 4k × 8 ROM (80C51)
  - 4k × 8 EPROM (87C51)
  - ROMless (80C31)
  - 128 × 8 RAM
  - Two 16-bit counter/timers
  - Full duplex serial channel
  - Boolean processor
- Memory addressing capability
  - 64k ROM and 64k RAM
- Power control modes:
  - Idle mode
  - Power-down mode
- CMOS and TTL compatible
- Five speed ranges at V_{CC} = 5V
  - 12MHz
  - 16MHz
  - 24MHz
  - 30MHz
  - 33MHz
- Five package styles
- Extended temperature ranges
- OTP package available

PIN CONFIGURATIONS





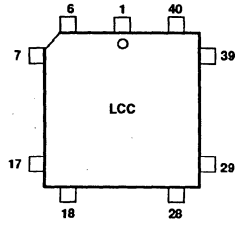


SEE PAGE 118 FOR QFP AND LCC PIN FUNCTIONS.

PART NUMBER SELECTION

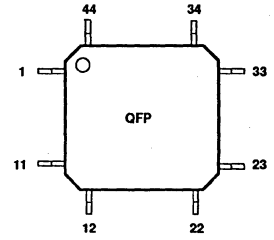
PHILIPS		SIGNETICS			TEMPERATURE °C	
ROMless	ROM	ROMless	ROM	EPROM	AND PACKAGE	SPEED
				SC87C51CCF40	0 to +70, CDIP	12MHz
				SC87C51CCK44	0 to +70, CLCC	12MHz
PCB80C31BH2-12P	PCB80C51BH-2P	SC80C31BCCN40	SC80C51BCCN40	SC87C51CCN40	0 to +70, PDIP	12MHz
PCB80C31BH2-12WP	PCB80C51BH-2WP	SC80C31BCCA44	SC80C51BCCA44	SC87C51CCA44	0 to +70, PLCC	12MHz
PCB80C31BH2-12H	PCB80C51BH-2H	SC80C31BCCB44	SC80C51BCCB44	SC87C51CCB44	0 to +70, PQFP	12MHz
				SC87C51ACF40	–40 to +85, CDIP	12MHz
				SC87C51ACK44	–40 to +85, CLCC	12MHz
		SC80C31BACN40	SC80C51BACN40	SC87C51ACN40	–40 to +85, PDIP	12MHz
		SC80C31BACA44	SC80C51BACA44	SC87C51ACA44	–40 to +85, PLCC	12MHz
		SC80C31BACB44	SC80C51BACB44	SC87C51ACB44	–40 to +85, PQFP	12MHz
				SC87C51CGF40	0 to +70, CDIP	16MHz
				SC87C51CGK44	0 to +70, CLCC	16MHz
PCB80C31BH3-16P	PCB80C51BH-3P	SC80C31BCGN40	SC80C51BCGN40	SC87C51CGN40	0 to +70, PDIP	16MHz
PCB80C31BH3-16WP	PCB80C51BH-3WP	SC80C31BCGA44	SC80C51BCGA44	SC87C51CGA44	0 to +70, PLCC	16MHz
PCB80C31BH3-16H	PCB80C51BH-3H	SC80C31BCGB44	SC80C51BCGB44	SC87C51CGB44	0 to +70, PQFP	16MHz
				SC87C51AGF40	–40 to +85, CDIP	16MHz
				SC87C51AGK44	–40 to +85, CLCC	16MHz
PCF80C31BH3-16P	PCF80C51BH-3P	SC80C31BAGN40	SC80C51BAGN40	SC87C51AGN40	–40 to +85, PDIP	16MHz
PCF80C31BH3-16WP	PCF80C51BH-3WP	SC80C31BAGA44	SC80C51BAGA44	SC87C51AGA44	–40 to +85, PLCC	16MHz
PCF80C31BH3-16H	PCF80C51BH-3H	SC80C31BAGB44	SC80C51BAGB44	SC87C51AGB44	–40 to +85, PQFP	16MHz
PCA80C31BH3-16P	PCA80C51BH-3P				–40 to +125, PDIP	16MHz
PCA80C31BH3-16WP	PCA80C51BH-3WP				–40 to +125, PLCC	16MHz
				SC87C51CPF40	0 to +70, CDIP	24MHz
				SC87C51CPK44	0 to +70, CLCC	24MHz
PCB80C31BH4-24P	PCB80C51BH-4P	SC80C31BCPN40	SC80C51BCPN40	SC87C51CPN40	0 to +70, PDIP	24MHz
PCB80C31BH4-24WP	PCB80C51BH-4WP	SC80C31BCPA44	SC80C51BCPA44	SC87C51CPA44	0 to +70, PLCC	24MHz
PCB80C31BH4-24H	PCB80C51BH-4H	SC80C31BCPB44	SC80C51BCPB44	SC87C51CPB44	0 to +70, PQFP	24MHz
				SC87C51APF40	–40 to +85, CDIP	24MHz
				SC87C51APK44	–40 to +85, CLCC	24MHz
PCF80C31BH4-24P	PCF80C51BH-4P	SC80C31BAPN40	SC80C51BAPN40	SC87C51APN40	–40 to +85, PDIP	24MHz
PCF80C31BH4-24WP	PCF80C51BH-4WP	SC80C31BAPA44	SC80C51BAPA44	SC87C51APA44	–40 to +85, PLCC	24MHz
PCF80C31BH4-24H	PCF80C51BH-4H	SC80C31BAPB44	SC80C51BAPB44	SC87C51APB44	–40 to +85, PQFP	24MHz
PCB80C31BH5-30P	PCB80C51BH-5P				0 to +70, PDIP	30MHz
PCB80C31BH5-30WP	PCB80C51BH-5WP				0 to +70, PLCC	30MHz
PCB80C31BH5-30H	PCB80C51BH-5H				0 to +70, PQFP	30MHz
				SC87C51CYF40	0 to +70, CDIP	33MHz
				SC87C51CYK44	0 to +70, CLCC	33MHz
		SC80C31BCYN40	SC80C51BCYN40	SC87C51CYN40	0 to +70, PDIP	33MHz
		SC80C31BCYA44	SC80C51BCYA44	SC87C51CYA44	0 to +70, PLCC	33MHz
		SC80C31BCYB44	SC80C51BCYB44	SC87C51CYB44	0 to +70, PQFP	33MHz
				SC87C51AYF40	–40 to +85, CDIP	33MHz
				SC87C51AYK44	–40 to +85, CLCC	33MHz
		SC80C31BAYN40	SC80C51BAYN40	SC87C51AYN40	–40 to +85, PDIP	33MHz
		SC80C31BAYA44	SC80C51BAYA44	SC87C51AYA44	–40 to +85, PLCC	33MHz
		SC80C31BAYB44	SC80C51BAYB44	SC87C51AYB44	–40 to +85, PQFP	33MHz

LCC PIN FUNCTIONS



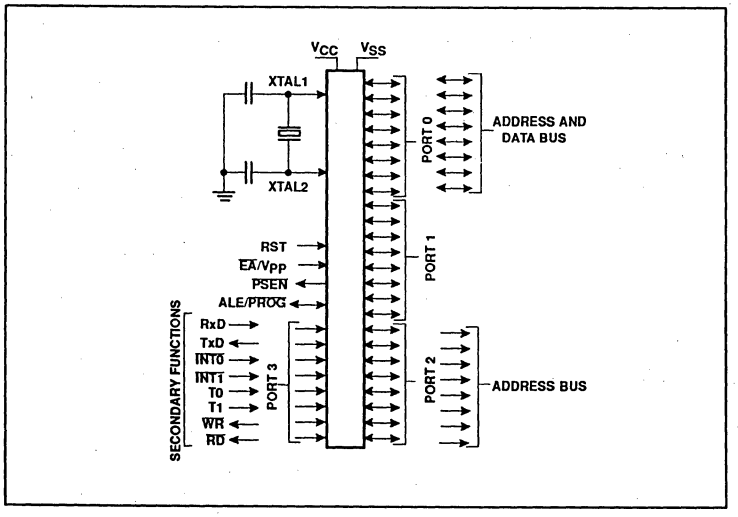
Pin	Function	Pin	Function
1	NC	23	NC
2	P1.0	24	P2.0/A8
3	P1.1	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6	30	P2.6/A14
9	P1.7	31	P2.7/A15
10	RST	32	$\overline{PSEN}$
11	P3.0/RxD	33	ALE/$\overline{PROG}$
12	NC	34	NC
13	P3.1/TxD	35	$\overline{EA}$/V_{PP}
14	P3.2/$\overline{INT0}$	36	P0.7/AD7
15	P3.3/$\overline{INT1}$	37	P0.6/AD6
16	P3.4/T0	38	P0.5/AD5
17	P3.5/T1	39	P0.4/AD4
18	P3.6/$\overline{WR}$	40	P0.3/AD3
19	P3.7/$\overline{RD}$	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	V_{SS}	44	V_{CC}

QFP PIN FUNCTIONS

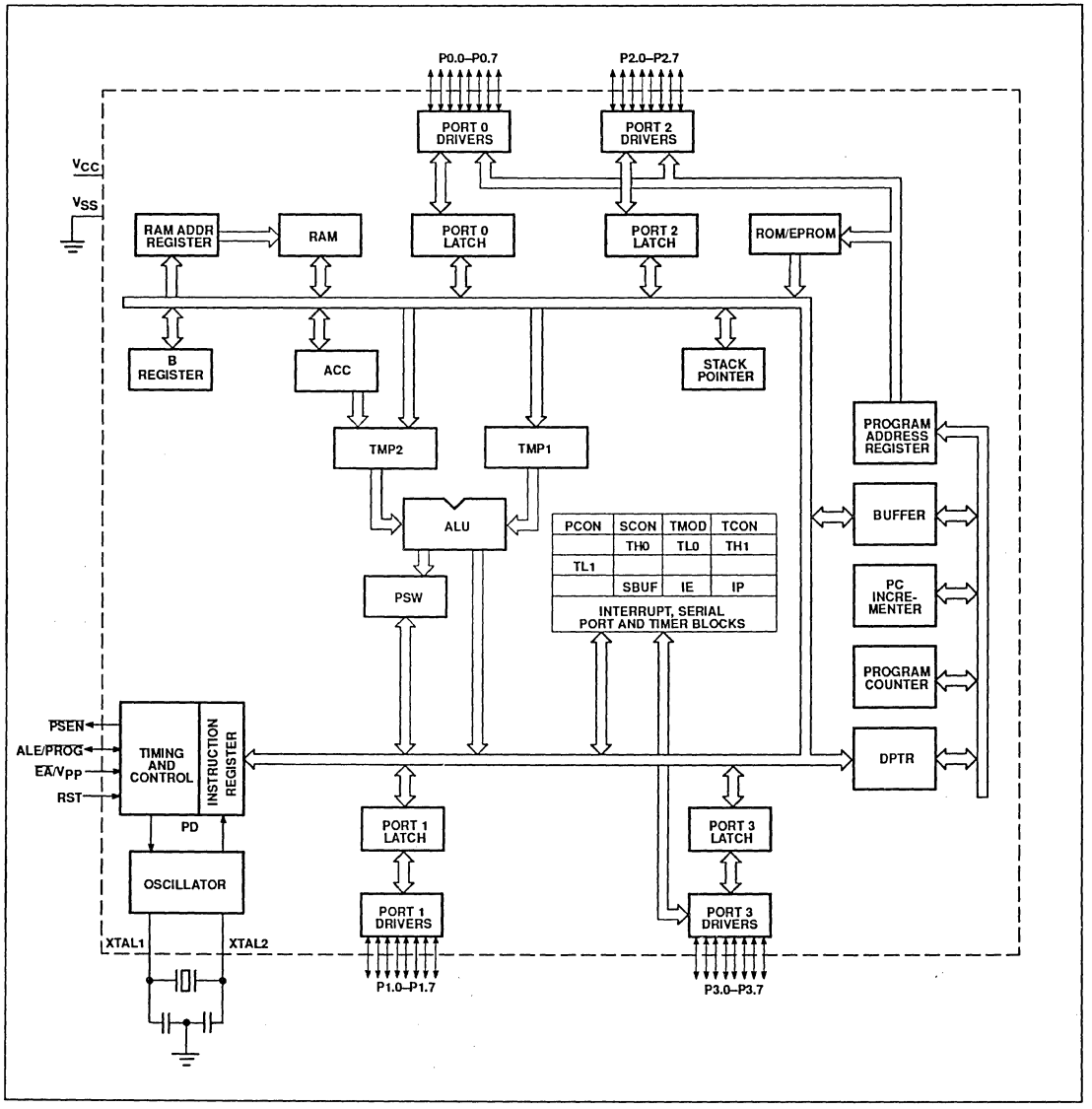


Pin	Function	Pin	Function
1	P1.5	23	P2.5/A13
2	P1.6	24	P2.6/A14
3	P1.7	25	P2.7/A15
4	RST	26	$\overline{PSEN}$
5	P3.0/RxD	27	ALE/$\overline{PROG}$
6	NC	28	NC
7	P3.1/TxD	29	$\overline{EA}$/V_{PP}
8	P3.2/$\overline{INT0}$	30	P0.7/AD7
9	P3.3/$\overline{INT1}$	31	P0.6/AD6
10	P3.4/T0	32	P0.5/AD5
11	P3.5/T1	33	P0.4/AD4
12	P3.6/$\overline{WR}$	34	P0.3/AD3
13	P3.7/$\overline{RD}$	35	P0.2/AD2
14	XTAL2	36	P0.1/AD1
15	XTAL1	37	P0.0/AD0
16	V_{SS}	38	V_{CC}
17	NC	39	NC
18	P2.0/A8	40	P1.0
19	P2.1/A9	41	P1.1
20	P2.2/A10	42	P1.2
21	P2.3/A11	43	P1.3
22	P2.4/A12	44	P1.4

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{SS}	20	22	16	I	**Ground:** 0V reference.
V_{CC}	40	44	38	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–36	37–30	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 also outputs the code bytes during program verification in the 87C51. External pull-ups are required during program verification.
P1.0–P1.7	1–8	2–9	40-44, 1–3	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 1 also receives the low-order address byte during program memory verification.
P2.0–P2.7	21–28	24–31	18–25	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	5, 7–13	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 80C51 family, as listed below:
	10	11	5	I	**RxD (P3.0):** Serial input port
	11	13	7	O	**TxD (P3.1):** Serial output port
	12	14	8	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	9	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	10	I	**T0 (P3.4):** Timer 0 external input
	15	17	11	I	**T1 (P3.5):** Timer 1 external input
	16	18	12	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	13	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	4	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE/$\overline{PROG}$	30	33	27	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{PSEN}$	29	32	26	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$/V_{PP}	31	35	29	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 0FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 0FFFH. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	19	21	15	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	14	O	**Crystal 2:** Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-up reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-up, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON.

Table 1 shows the state of I/O ports during low current operating modes.

ROM CODE SUBMISSION

When submitting ROM code for the 80C51, the following must be specified:

1. 4k byte user ROM data
2. 32 byte ROM encryption key (SC80C51 only)
3. ROM security bits (SC80C51 only).

ADDRESS	CONTENT	BIT(S)	COMMENT
0000H to 0FFFH	DATA	7:0	User ROM Data
1000H to 101FH	KEY	7:0	ROM Encryption Key
1020H	SEC	0	ROM Security Bit 1
1020H	SEC	1	ROM Security Bit 2

Security Bit 1: When programmed, this bit has two effects on masked ROM parts:
1. External MOVC is disabled, and
2. EA# is latched on Reset.

Security Bit 2: When programmed, this bit inhibits Verify User ROM.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	PSEN	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Electrical Deviations from Commercial Specifications for Extended Temperature Range (87C51)
DC and AC parameters not included here are the same as in the commercial temperature range table.

DC ELECTRICAL CHARACTERISTICS

T_{amb} = –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (Signetics Parts Only)
T_{amb} = –40°C to +125°C, V_{CC} = 5V ±10%, V_{SS} = 0V (Philips Parts Only)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNIT
			MIN	MAX	
V_{IL}	Input low voltage, except $\overline{EA}$ (Signetics)		–0.5	$0.2V_{CC}$–0.15	V
V_{IL}	Input low voltage, except $\overline{EA}$ (Philips)		–0.5	$0.2V_{CC}$–0.25	V
V_{IL1}	Input low voltage to $\overline{EA}$		–0.5	$0.2V_{CC}$–0.45	V
V_{IH}	Input high voltage, except XTAL1, RST		$0.2V_{CC}$+1	V_{CC}+0.5	V
V_{IH1}	Input high voltage to XTAL1, RST		$0.7V_{CC}$+0.1	V_{CC}+0.5	V
I_{IL}	Logical 0 input current, ports 1, 2, 3	V_{IN} = 0.45V		–75	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3	V_{IN} = 2.0V		–750	µA
I_{CC}	Power supply current: Active mode[1] @ 16MHz (Philips) Active mode @ 12MHz (Signetics) Idle mode[2] @ 16MHz (Philips) Idle mode @ 12MHz (Signetics) Power-down mode[3] (Philips) Power-down mode (Signetics)	V_{CC} = 4.5–5.5V		 25 20 6.5 5 75 50	 mA mA mA mA µA µA

NOTES:
1. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f$ = 10ns; $V_{IL} = V_{SS}$ + 0.5V; $V_{IH} = V_{CC}$ – 0.5V; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = V_{CC}.
2. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f$ = 10ns; $V_{IL} = V_{SS}$ + 0.5V; $V_{IH} = V_{CC}$ – 0.5V; XTAL2 not connected; $\overline{EA}$ = Port 0 = V_{CC}; RST = V_{SS}.
3. The power-down current is measured with all output pins disconnected, XTAL2 not connected, $\overline{EA}$ = Port 0 = V_{CC}; RST = V_{SS}.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Operating temperature under bias	0 to +70 or –40 to +85	°C
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}$/V_{PP} pin to V_{SS}	0 to +13.0	V
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Maximum I_{OL} per I/O pin	15	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.5	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V (80C31/51)
T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (87C51)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			MIN	TYPICAL[1]	MAX	
V_{IL}	Input low voltage, except $\overline{EA}$[7]		–0.5		$0.2V_{CC}$–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$[7]		0		$0.2V_{CC}$–0.3	V
V_{IH}	Input high voltage, except XTAL1, RST[7]		$0.2V_{CC}$+0.9		V_{CC}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST[7]		$0.7V_{CC}$		V_{CC}+0.5	V
V_{OL}	Output low voltage, ports 1, 2, 3[11]	I_{OL} = 1.6mA[2]			0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$[11]	I_{OL} = 3.2mA[2]			0.45	V
V_{OH}	Output high voltage, ports 1, 2, 3, ALE, $\overline{PSEN}$[3]	I_{OH} = –60µA, I_{OH} = –25µA I_{OH} = –10µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
V_{OH1}	Output high voltage (port 0 in external bus mode)	I_{OH} = –800µA, I_{OH} = –300µA I_{OH} = –80µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
I_{IL}	Logical 0 input current, ports 1, 2, 3[7]	V_{IN} = 0.45V			–50	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3[7]	See note 4			–650	µA
I_{LI}	Input leakage current, port 0	V_{IN} = V_{IL} or V_{IH}			±10	µA
I_{CC}	Power supply current:[7] Active mode @ 12MHz[8] (Philips) Active mode @ 12MHz[5] (Signetics) Idle mode @ 12MHz[9] (Philips) Idle mode @ 12MHz (Signetics) Power-down mode[10] (Philips and Signetics)	See note 6		 11.5 1.3 3	 18 19 4.4 4 50	 mA mA mA mA µA
R_{RST}	Internal reset pull-down resistor (Signetics) (Philips)		50 50		300 150	kΩ kΩ
C_{IO}	Pin capacitance				10	pF

NOTES:

1. Typical ratings are not guaranteed. The values listed are at room temperature, 5V.
2. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.. I_{OL} can exceed these conditions provided that no single output sinks more than 5mA and no more than two outputs exceed the test conditions.
3. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
4. Pins of ports 1, 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
5. I_{CC}MAX at other frequencies (for Signetics parts) is given by: Active mode: I_{CC}MAX = 1.43 X FREQ + 1.90: Idle mode: I_{CC}MAX = 0.14 X FREQ +2.31, where FREQ is the external oscillator frequency in MHz. I_{CC}MAX is given in mA. See Figure 7.
6. See Figures 8 through 11 for I_{CC} test conditions.
7. For Signetics parts when T_{amb} = –40°C to +85°C or Philips parts when T_{amb} = –40°C to +125°C, see DC Electrical Characteristics table on previous page.
8. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f$ = 10ns; $V_{IL} = V_{SS}$ + 0.5V; $V_{IH} = V_{CC}$ – 0.5V; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = V_{CC}.
9. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f$ = 10ns; $V_{IL} = V_{SS}$ + 0.5V; $V_{IH} = V_{CC}$ – 0.5V; XTAL2 not connected; $\overline{EA}$ = Port 0 = V_{CC}; RST = V_{SS}.
10. The power-down current is measured with all output pins disconnected, XTAL2 not connected, $\overline{EA}$ = Port 0 = V_{CC}; RST = V_{SS}.
11. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
    Maximum I_{OL} per port pin: 15mA
    Maximum I_{OL} per 8-bit port: 26mA
    Maximum I_{OL} total for all outputs: 67mA
    If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

AC ELECTRICAL CHARACTERISTICS FOR 12–33MHz SIGNETICS DEVICES

T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V (80C31/51)[1, 2, 4]
T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (87C51)

SYMBOL	FIGURE	PARAMETER	VARIABLE CLOCK[3] MIN	VARIABLE CLOCK[3] MAX	UNIT
$1/t_{CLCL}$		Oscillator frequency: **Speed Versions**			
		SC80C31/51 C	3.5	12	MHz
		G	3.5	16	MHz
		P	3.5	24	MHz
		Y	3.5	33	MHz
t_{LHLL}	1	ALE pulse width	$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	$t_{CLCL}-13$		ns
t_{LLAX}	1	Address hold after ALE low	$t_{CLCL}-20$		ns
t_{LLIV}	1	ALE low to valid instruction in		$4t_{CLCL}-65$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	$t_{CLCL}-13$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	$3t_{CLCL}-20$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		$3t_{CLCL}-45$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		$t_{CLCL}-10$	ns
t_{AVIV}	1	Address to valid instruction in		$5t_{CLCL}-55$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10	ns
Data Memory					
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		$5t_{CLCL}-90$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		$2t_{CLCL}-28$	ns
t_{LLDV}	2, 3	ALE low to valid data in		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	$4t_{CLCL}-75$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	$t_{CLCL}-20$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	$t_{CLCL}-20$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	$t_{CLCL}-20$	$t_{CLCL}+25$	ns
External Clock					
t_{CHCX}	4	High time	12		ns
t_{CLCX}	4	Low time	12		ns
t_{CLCH}	4	Rise time		20	ns
t_{CHCL}	4	Fall time		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. For all Signetics speed versions only.
4. Interfacing the 80C31/51 to devices with float times up to 30ns is permitted. This limited bus contention will not cause damage to port 0 drivers.

AC ELECTRICAL CHARACTERISTICS FOR PHILIPS DEVICES

T_{amb} = 0°C to +70°C, V_{CC} = 5V ±20%, V_{SS} = 0V (80C31/51)[1, 2, 4]

SYMBOL	FIGURE	PARAMETER	VARIABLE CLOCK[3] MIN	VARIABLE CLOCK[3] MAX	UNIT
$1/t_{CLCL}$		Oscillator frequency: **Speed Versions** PCA/PCB/PCF80C31/51 –3 PCB/PCF80C31/51 –4 PCB/FB80C31/51 –5	 1.2 1.2 1.2	 16 24 30	 MHz MHz MHz
t_{LHLL}	1	ALE pulse width	$2t_{CLCL}$–40		ns
t_{AVLL}	1	Address valid to ALE low	t_{CLCL}–25		ns
t_{LLAX}	1	Address hold after ALE low	t_{CLCL}–25		ns
t_{LLIV}	1	ALE low to valid instruction in		$4t_{CLCL}$–65	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	t_{CLCL}–25		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	$3t_{CLCL}$–45		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		$3t_{CLCL}$–60	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		t_{CLCL}–25	ns
t_{AVIV}	1	Address to valid instruction in		$5t_{CLCL}$–80	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10	ns
Data Memory					
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	$6t_{CLCL}$–100		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	$6t_{CLCL}$–100		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		$5t_{CLCL}$–90	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		$2t_{CLCL}$–28	ns
t_{LLDV}	2, 3	ALE low to valid data in		$8t_{CLCL}$–150	ns
t_{AVDV}	2, 3	Address to valid data in		$9t_{CLCL}$–165	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	$3t_{CLCL}$–50	$3t_{CLCL}$+50	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	$4t_{CLCL}$–75		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	t_{CLCL}–30		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	t_{CLCL}–25		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	t_{CLCL}–25	t_{CLCL}+25	ns
External Clock					
t_{CHCX}	4	High time	15		ns
t_{CLCX}	4	Low time	15		ns
t_{CLCH}	4	Rise time		20	ns
t_{CHCL}	4	Fall time		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. For all Philips speed versions only.
4. Interfacing the 80C31/51 to devices with float times up to 30ns is permitted. This limited bus contention will not cause damage to port 0 drivers.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

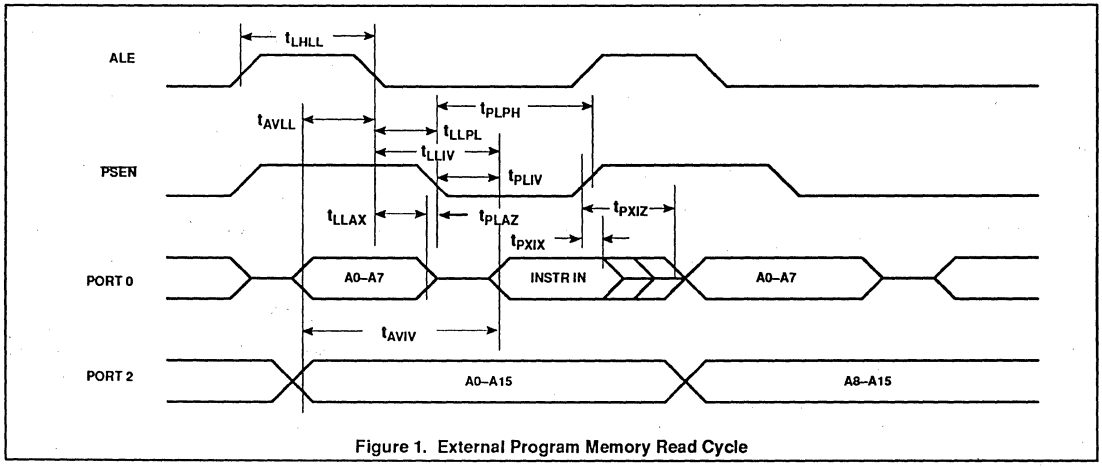


Figure 1. External Program Memory Read Cycle

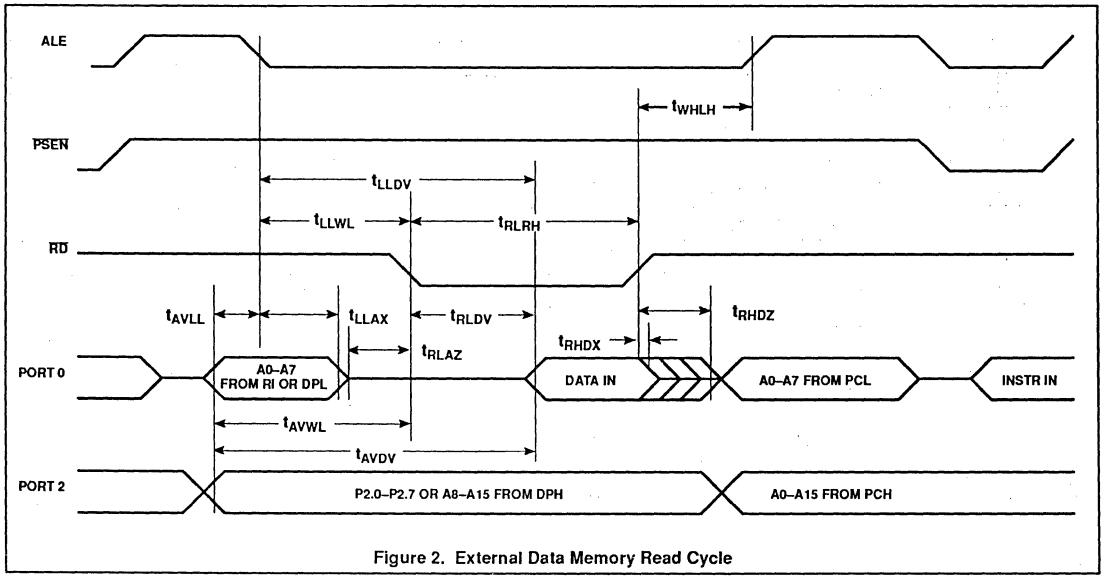


Figure 2. External Data Memory Read Cycle

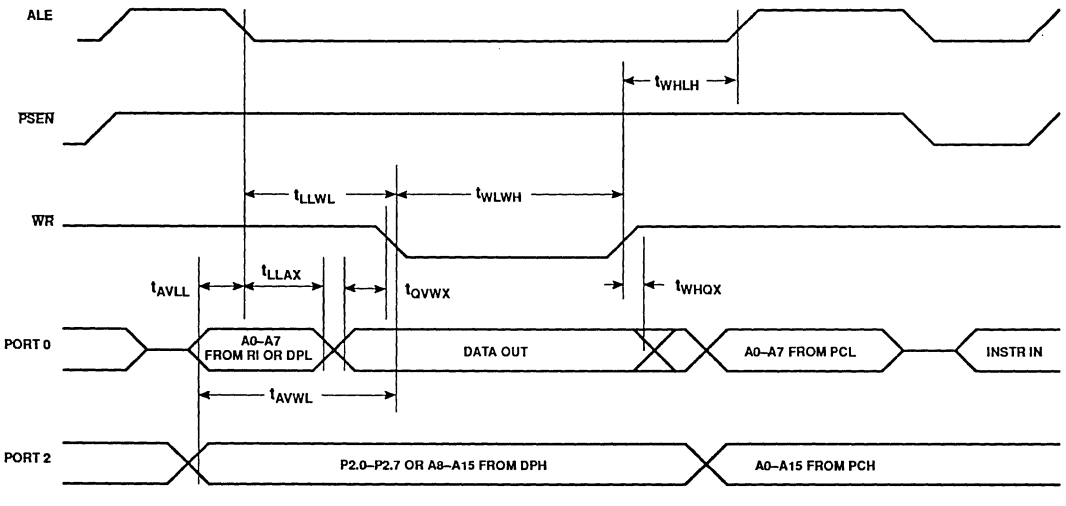


Figure 3. External Data Memory Write Cycle

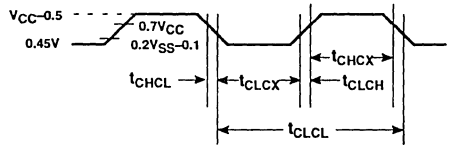
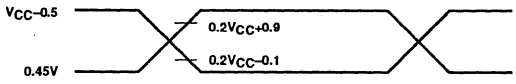
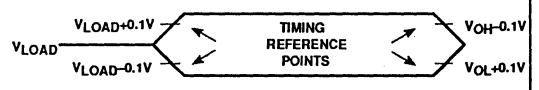


Figure 4. External Clock Drive



NOTE:
AC inputs during testing are driven at V_{CC} –0.5 for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at V_{IH} min for a logic '1' and V_{IL} for a logic '0'.

Figure 5. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 6. Float Waveform

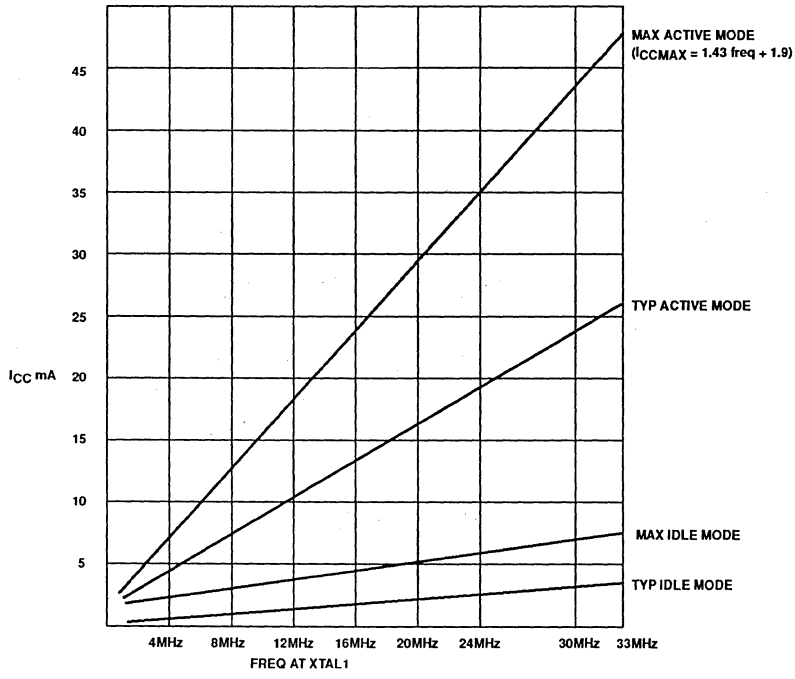


Figure 7. I_{CC} vs. FREQ
Valid only within frequency specifications of the device under test

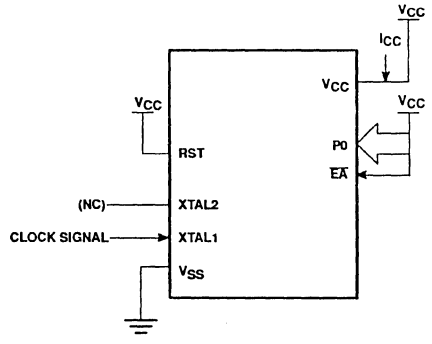


Figure 8. I_{CC} Test Condition, Active Mode
All other pins are disconnected

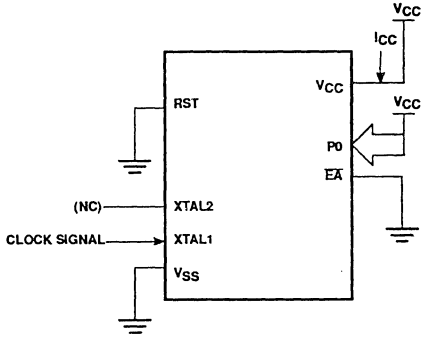


Figure 9. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

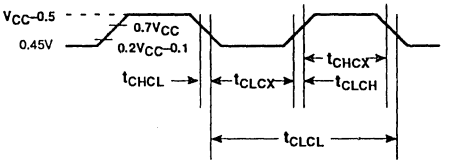


Figure 10. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

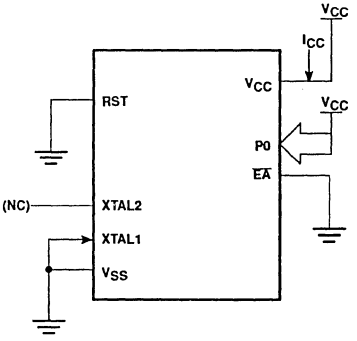


Figure 11. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

EPROM CHARACTERISTICS

The 87C51 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C51 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C51 manufactured by Philips Corporation.

Table 2 shows the logic levels for reading the signature bytes, and for programming the program memory, the encryption table, and the security bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 12 and 13. Figure 14 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 12. Note that the 87C51 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 12. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 2 and 3 specified in Table 2 are held at the 'Program Code Data' levels indicated in Table 2. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 13.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 1FH, using the 'Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the security bits, repeat the 25 pulse programming sequence using the 'Pgm Security Bit' levels. After one security bit is programmed, further programming of the code memory and encryption table is disabled. However, the other secrity bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

PROGRAM VERIFICATION

If security bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 1 and 2 as shown in Figure 14. The other pins are held at the 'Verify Code Data' levels indicated in Table 2. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(031H) = 92H indicates 87C51

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 2, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345-5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 2. EPROM Progamming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm security bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm security bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:
1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.

* ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

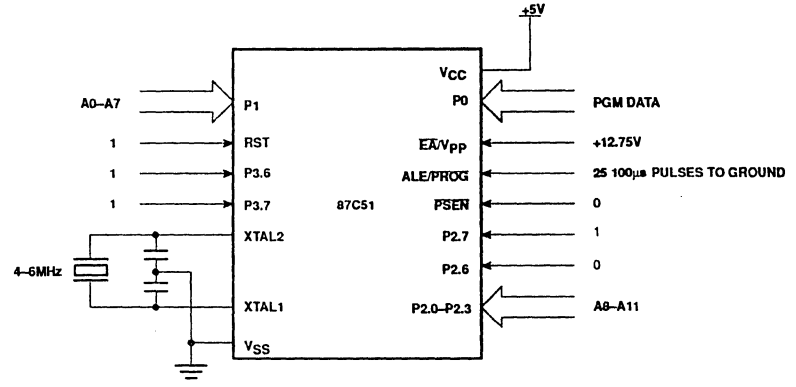


Figure 12. Programming Configuration

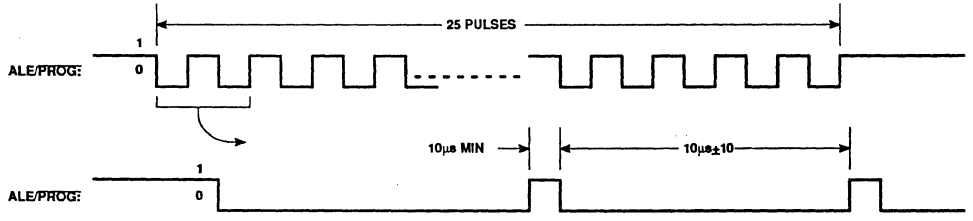


Figure 13. PROG Waveform

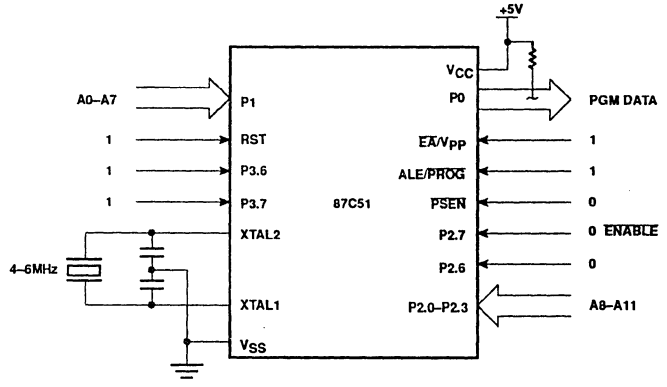


Figure 14. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_{amb} = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 15)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 (ENABLE) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		us
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		us
t_{GLGH}	$\overline{PROG}$ width	90	110	us
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		us

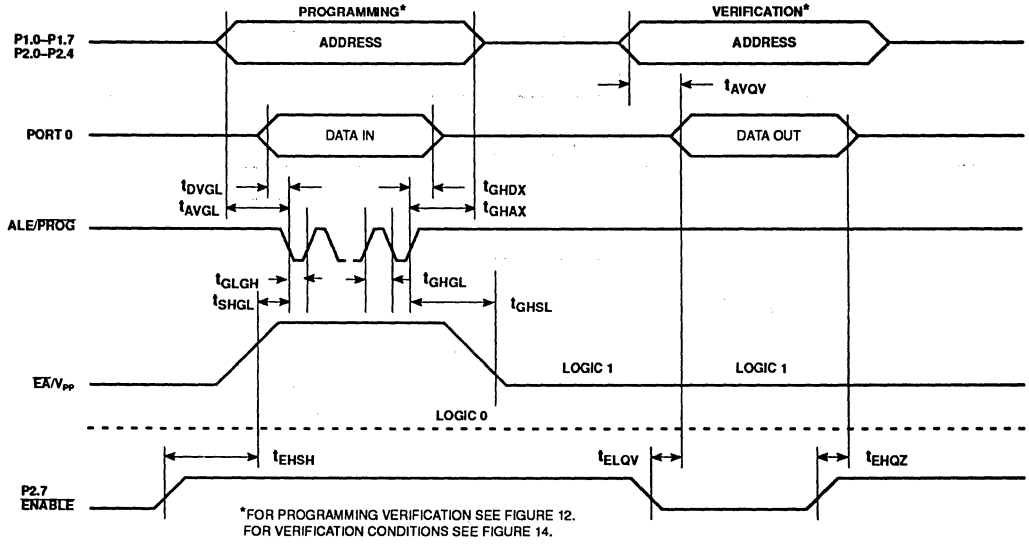


Figure 15. EPROM Programming and Verification

Section 2
Inter-Integrated (I^2C) Circuit Bus

CONTENTS

I^2C Bus Specification 134

I^2C Address Allocation Table 154

Assigned I^2C Bus Addresses 155

I^2C Peripheral Selection Guide 156

I²C-bus specification (including fast-mode)

PREFACE

This specification is an updated version including the following latest modifications:

- Programming of a slave address by software has been omitted. The realization of this feature is rather complicated and has not been used.
- The 'low-speed mode' has been omitted. This mode is, in fact, a subset of the total I²C-bus specification and need not be specified explicitly.
- The 'fast-mode' is added. This allows a fourfold increase of bit rate up to 400 kbit/s. Fast-mode devices are downwards compatible i.e. they can be used in a 0 to 100 kbit/s I²C-bus system.
- 10-bit addressing is added. This allows 1024 additional slave addresses.
- Slope control and input filtering for fast-mode devices is specified to improve the EMC behaviour.

NOTE

Neither the 100 kbit/s I²C-bus system nor the 100 kbit/s devices have been changed.

I²C-bus specification (including fast-mode)

1.0 INTRODUCTION

For 8-bit applications, such as those requiring single-chip microcontrollers, certain design criteria can be established:

- A complete system usually consists of at least one microcontroller and other peripheral devices such as memories and I/O expanders.
- The cost of connecting the various devices within the system must be minimized.
- Such a system usually performs a control function and doesn't require high-speed data transfer.
- Overall efficiency depends on the devices chosen and the interconnecting bus structure.

In order to produce a system to satisfy these criteria, a serial bus structure is needed. Although serial buses don't have the throughput capability of parallel buses, they do require less wiring and fewer connecting pins. However, a bus is not merely an interconnecting wire, it embodies all the formats and procedures for communication within the system.

Devices communicating with each other on a serial bus must have some form of protocol which avoids all possibilities of confusion, data loss and blockage of information. Fast devices must be able to communicate with slow devices. The system must not be dependent on the devices connected to it, otherwise modifications or improvements would be impossible. A procedure has also to be devised to decide which device will be in control of the bus and when. And, if different devices with different clock speeds are connected to the bus, the bus clock source must be defined. All these criteria are involved in the specification of the I²C-bus.

2.0 THE I²C-BUS CONCEPT

Any IC fabrication process (NMOS, CMOS, bipolar) can be supported by the I²C-bus. Two wires, serial data (SDA) and serial clock (SCL), carry information between the devices connected to the bus. Each device is recognised by a unique address - whether it's a microcontroller, LCD driver, memory or keyboard interface - and can operate as either a transmitter or receiver, depending on the function of the device. Obviously an LCD driver is only a receiver, whereas a memory can both receive and transmit data. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers (see Table 1). A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

The I²C-bus is a multi-master bus. This means that more than one device capable of controlling the bus can be connected to it. As masters are usually micro-controllers, let's consider the case of a data transfer between two microcontrollers connected to the I²C-bus (Fig.1). This highlights the master-slave and receiver-transmitter relationships to be found on the I²C-bus. It should be noted that these relationships are

Table 1 Definition of I²C-bus terminology

Term	Description
Transmitter	The device which sends the data to the bus
Receiver	The device which receives the data from the bus
Master	The device which initiates a transfer, generates clock signals and terminates a transfer
Slave	The device addressed by a master
Multi-master	More than one master can attempt to control the bus at the same time without corrupting the message
Arbitration	Procedure to ensure that, if more than one master simultaneously tries to control the bus, only one is allowed to do so and the message is not corrupted
Synchronization	Procedure to synchronize the clock signals of two or more devices

I²C-bus specification (including fast-mode)

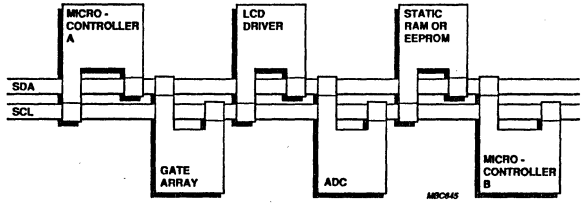


Fig.1 Example of an I²C-bus configuration

not permanent, but only depend on the direction of data transfer at that time. The transfer of data would proceed as follows:

1) Suppose microcontroller A wants to send information to microcontroller B:

- microcontroller A (master), addresses microcontroller B (slave)
- microcontroller A (master-transmitter), sends data to microcontroller B (slave-receiver)
- microcontroller A terminates the transfer.

2) If microcontroller A wants to receive information from microcontroller B:

- microcontroller A (master) addresses microcontroller B (slave)
- microcontroller A (master-receiver) receives data from microcontroller B (slave-transmitter)
- microcontroller A terminates the transfer.

Even in this case, the master (microcontroller A) generates the timing and terminates the transfer.

The possibility of connecting more than one microcontroller to the I²C-bus means that more than one master could try to initiate a data transfer at the same time. To avoid the chaos that might ensue from such an event - an arbitration procedure has been developed. This procedure relies on the wired-AND connection of all I²C interfaces to the I²C-bus.

If two or more masters try to put information onto the bus, the first to produce a 'one' when the other produces a 'zero' will lose the arbitration. The clock signals during arbitration are a synchronised combination of the clocks generated by the masters using the wired-AND connection to the SCL line (for more detailed information concerning arbitration see section 6.0).

Generation of clock signals on the I²C-bus is always the responsibility of master devices; each master generates its own clock signals when transferring data on the bus. Bus clock signals from a master can only be altered when they are stretched by a slow-slave device holding-down the clock line, or by another master when arbitration occurs.

3.0 GENERAL CHARACTERISTICS

Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via a pull-up resistor (see Fig.2). When the bus is free, both lines are HIGH. The output stages of devices connected to the bus must have an open-drain or open-collector in order to perform the wired-AND function. Data on the I²C-bus can be transferred at a rate up to 100 kbit/s in the standard-mode, or up to 400 kbit/s in the fast-mode. The number of interfaces connected to the bus is solely dependent on the limiting bus capacitance of 400 pF.

4.0 BIT TRANSFER

Due to the variety of different technology devices (CMOS, NMOS, bipolar) which can be connected to the I²C-bus, the levels of the logical '0' (LOW) and '1' (HIGH) are not fixed and depend on the associated level of V_{DD} (see Section 15.0 for

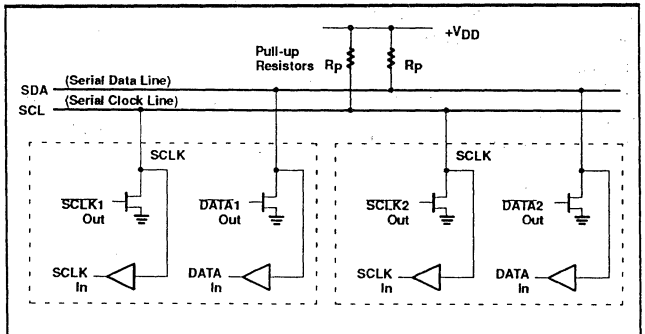


Fig.2 Connection of I²C-bus devices to the I²C-bus

I²C-bus specification (including fast-mode)

Electrical specifications). One clock pulse is generated for each data bit transferred.

4.1 Data validity

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the data line can only change when the clock signal on the SCL line is LOW (Fig.3).

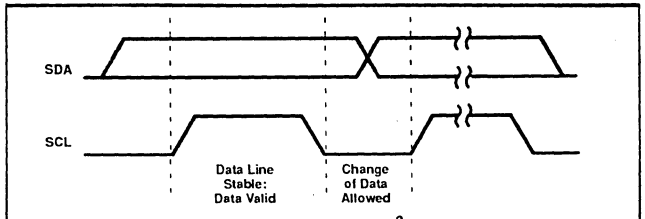


Fig.3 Bit transfer on the I²C-bus

4.2 START and STOP conditions

Within the procedure of the I²C-bus, unique situations arise which are defined as START and STOP conditions (see Fig.4).

A HIGH to LOW transition of the SDA line while SCL is HIGH is one such unique case. This situation indicates a START condition.

A LOW to HIGH transition of the SDA line while SCL is HIGH defines a STOP condition.

START and STOP conditions are always generated by the master. The bus is considered to be busy after the START condition. The bus is considered to be free again a certain time after the STOP condition. This bus free situation will be specified later (in Section 15.0).

Detection of START and STOP conditions by devices connected to the bus is easy if they incorporate the necessary interfacing hardware. However,

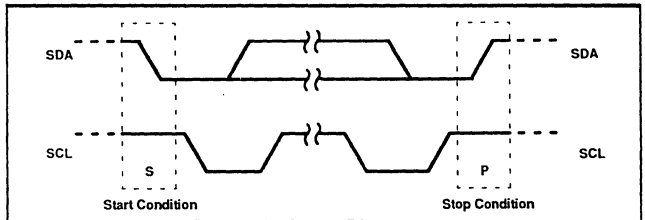


Fig.4 START and STOP conditions

microcontrollers with no such interface have to sample the SDA line at least twice per clock period in order to sense the transition.

5.0 TRANSFERRING DATA

5.1 Byte format

Every byte put on the SDA line must be 8-bits long. The number of bytes that can be transmitted per transfer is unrestricted. Each byte has to be followed by an acknowledge bit. Data is transferred with the most significant bit (MSB) first (Fig.5). If a receiver can't receive another complete byte of data until it has performed some other function, for example, servicing an internal interrupt, it can hold the clock line SCL LOW to force the transmitter into a wait state. Data transfer then continues when the receiver is ready for another byte of data and releases clock line SCL.

In some cases, it's permitted to use a different format from the I²C-bus format (for CBUS compatible devices for example). A message which starts with such an address can be terminated by generation of a STOP condition, even during the transmission of a byte. In this case, no acknowledge is generated (see section 8.1.3).

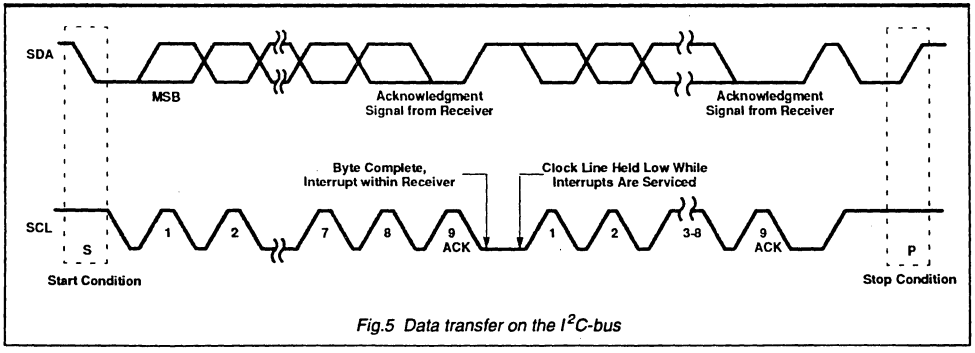


Fig.5 Data transfer on the I²C-bus

5.2 Acknowledge

Data transfer with acknowledge is obligatory. The acknowledge-related clock pulse is generated by the master. The transmitter releases the SDA line (HIGH) during the acknowledge clock pulse.

The receiver has to pull down the SDA line during the acknowledge clock pulse so that it remains stable LOW during the high period of this clock pulse (Fig.6). Of course, set-up and hold times must also be taken into account and these will be described in Section 15.0.

Usually, a receiver which has been addressed is obliged to generate an acknowledge after each byte has been received (except when the message starts with a CBUS address - see section 8.1.3).

When a slave-receiver doesn't acknowledge on the slave address (for example, it's unable to receive because it's performing some real-time function), the data line has to be left HIGH by the slave. The master can then generate a STOP condition to abort the transfer.

If a slave-receiver does acknowledge the slave address but, some time later in the transfer cannot receive any more data bytes, the master must again abort the transfer. This is indicated by the slave generating the not acknowledge on the first byte to follow. The slave leaves the data line HIGH and the master generates the STOP condition.

If a master-receiver is involved in a transfer, it must signal the end of data to the slave-transmitter by not generating an acknowledge on the last byte that was clocked out of the slave. The slave-transmitter must release the data line to allow the master to generate the STOP condition.

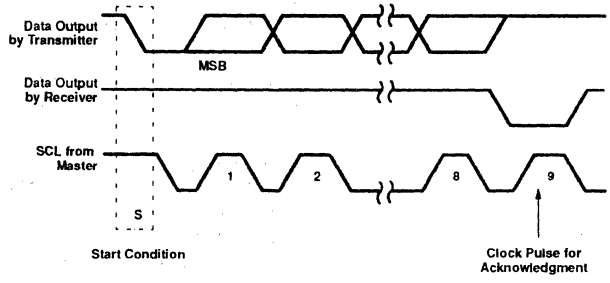


Fig.6 Acknowledge on the I²C-bus

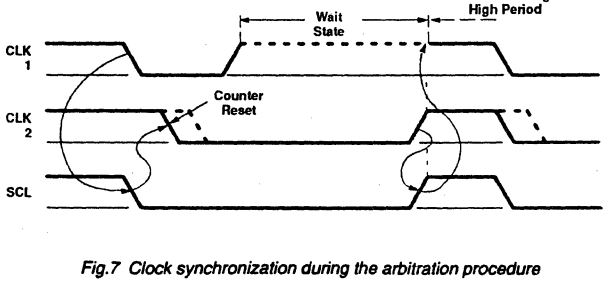


Fig.7 Clock synchronization during the arbitration procedure

6.0 ARBITRATION AND CLOCK GENERATION

6.1 Synchronization

All masters generate their own clock on the SCL line to transfer messages on the I²C-bus. Data is only valid during the clock HIGH period. A defined clock is therefore needed for the bit-by-bit arbitration procedure to take place.

Clock synchronization is performed using the wired-AND connection of I²C interfaces to the SCL line. This means that a HIGH to LOW transition on the SCL line will cause the devices concerned to start counting off their LOW period and, once a device clock has gone LOW, it will hold the SCL line in that state until the clock HIGH state is reached (Fig.7). However, the LOW to HIGH transition of this clock may not change the state of the SCL line if another clock is still within its LOW period. Therefore, the SCL line will be held LOW by the device with the longest LOW period. Devices with shorter LOW periods enter a HIGH wait-state during this time.

When all devices concerned have counted off their LOW period, the clock line will be released and go HIGH. There will then be no difference between the device clocks and the state of the SCL line and all the devices will start counting their HIGH periods. The first device to complete its HIGH period will again pull the SCL line LOW.

In this way, a synchronised SCL clock is generated with its LOW period determined by the device with the longest clock LOW period, and its HIGH period

determined by the one with the shortest clock HIGH period.

6.2 Arbitration

A master may start a transfer only if the bus is free. Two or more masters may generate a START condition within the minimum hold time ($t_{HD;STA}$) of the START condition which results in a defined START condition to the bus.

Arbitration takes place on the SDA line, while the SCL line is at the HIGH level, in such a way that the master which transmits a HIGH level, while another master is transmitting a LOW level will switch off its DATA output stage because the level on the bus doesn't correspond to its own level.

Arbitration can continue for many bits. Its first stage is comparison of the address bits (addressing information is in Sections 8.0 and 12.0). If the masters are each trying to address the same device, arbitration continues with comparison of the data. Because address and data information on the I²C-bus is used for arbitration, no information is lost during this process.

A master which loses the arbitration can generate clock pulses until the end of the byte in which it loses the arbitration.

If a master also incorporates a slave function and it loses arbitration during the addressing stage, it's possible that the winning master is trying to address it. The losing master must therefore switch over immediately to its slave-receiver mode.

Figure 8 shows the arbitration procedure for two masters. Of course, more may be involved (depending on how many masters are connected to the bus). The moment there is a difference between the internal data level of the master generating DATA 1 and the actual level on the SDA line, its data output is switched off, which means that a HIGH output level is then connected to the bus. This will not affect the data transfer initiated by the winning master. Since control of the I²C-bus is decided solely on the address and data sent by competing masters, there is no central master, nor any order of priority on the bus.

Special attention must be paid if, during a serial transfer, the arbitration procedure is still in progress at the moment when a repeated START condition or a STOP condition is transmitted to the I²C-bus. If it's possible for such a situation to occur, the masters involved must send this repeated START condition or STOP condition at the same position in the format frame. In other words, arbitration isn't allowed between:

- A repeated START condition and a data bit
- A STOP condition and a data bit
- A repeated START condition and a STOP condition.

6.3 Use of the clock synchronising mechanism as a handshake

In addition to being used during the arbitration procedure, the clock synchronization mechanism can be used to enable receivers to cope with fast data transfers, on either a byte level or a bit level.

On the byte level, a device may be able to receive bytes of data at a fast rate, but needs more time to store a received byte or prepare another byte to be transmitted. Slaves can then hold the SCL line LOW after reception and acknowledgement of a byte to force the master into a wait state until the slave is ready for the next byte transfer in a type of handshake procedure.

On the bit level, a device such as a microcontroller without, or with only a limited hardware I²C interface on-chip can slow down the bus clock by extending each clock LOW period. In this way, the speed of any master is adapted to the internal operating rate of this device.

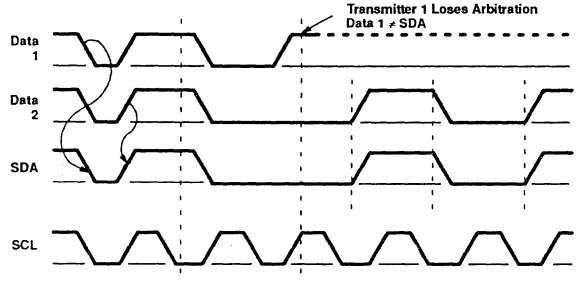


Fig.8 Arbitration procedure of two masters

I²C-bus specification (including fast-mode)

7.0 FORMATS WITH 7-BIT ADDRESSES

Data transfers follow the format shown in Fig.9. After the START condition (S), a slave address is sent. This address is 7 bits long followed by an eighth bit which is a data direction bit (R/$\overline{W}$) - a 'zero' indicates a transmission (WRITE), a 'one' indicates a request for data (READ). A data transfer is always terminated by a STOP condition (P) generated by the master. However, if a master still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another slave without first generating a STOP condition. Various combinations of read/write formats are then possible within such a transfer.

Possible data transfer formats are:

- Master-transmitter transmits to slave-receiver. The transfer direction is not changed (Fig.10).

- Master reads slave immediately after first byte (Fig.11). At the moment of the first acknowledge, the master-transmitter becomes a master-receiver and the slave-receiver becomes a slave-transmitter. This acknowledge is still generated by the slave. The STOP condition is generated by the master.

- Combined format (Fig.12). During a change of direction within a transfer, the START condition and the slave address are both repeated, but with the R/$\overline{W}$ bit reversed.

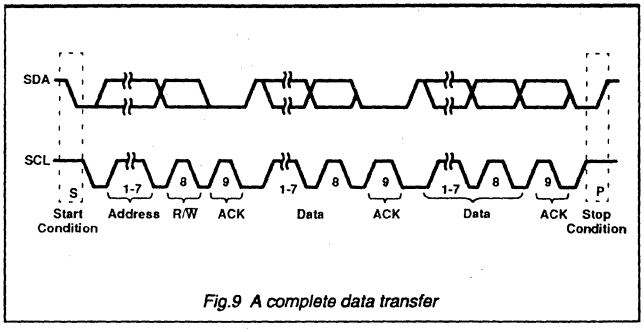


Fig.9 A complete data transfer

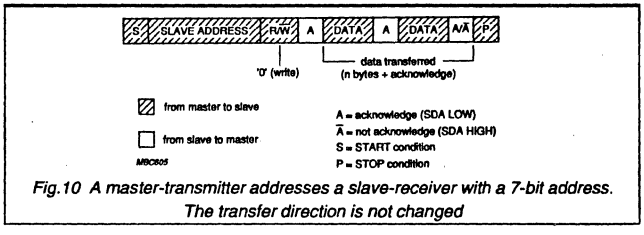


Fig.10 A master-transmitter addresses a slave-receiver with a 7-bit address. The transfer direction is not changed

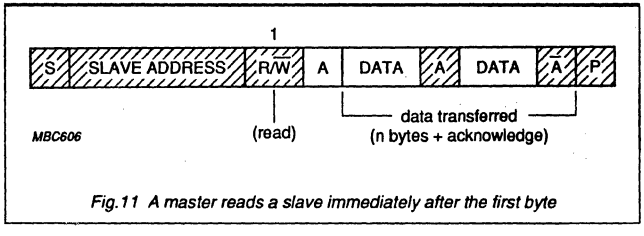


Fig.11 A master reads a slave immediately after the first byte

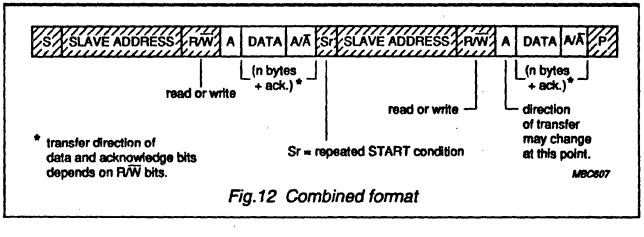


Fig.12 Combined format

NOTES:

1) Combined formats can be used, for example, to control a serial memory. During the first data byte, the internal memory location has to be written. After the START condition and slave address is repeated, data can be transferred.
2) All decisions on auto-increment or decrement of previously accessed memory locations etc. are taken by the designer of the device.
3) Each byte is followed by an acknowledgement bit as indicated by the A or $\overline{A}$ blocks in the sequence.
4) I²C-bus compatible devices must reset their bus logic on receipt of a START or repeated START condition such that they all anticipate the sending of a slave address.

I²C-bus specification (including fast-mode)

8.0 7-BIT ADDRESSING (see section 13 for 10-bit addressing)

The addressing procedure for the I²C-bus is such that the first byte after the START condition usually determines which slave will be selected by the master. The exception is the 'general call' address which can address all devices. When this address is used, all devices should, in theory, respond with an acknowledge. However, devices can be made to ignore this address. The second byte of the general call address then defines the action to be taken. This procedure is explained in more detail in Section 8.1.1.

8.1 Definition of bits in the first byte

The first seven bits of the first byte make up the slave address (Fig.13). The eighth bit is the LSB (least significant bit). It determines the direction of the message. A 'zero' in the least significant position of the first byte means that the master will write information to a selected slave. A 'one' in this position means that the master will read information from the slave.

When an address is sent, each device in a system compares the first 7 bits after the START condition with its address. If they match, the device considers itself addressed by the master as a slave-receiver or slave-transmitter, depending on the R/$\overline{W}$ bit.

A slave address can be made-up of a fixed and a programmable part. Since it's likely that there will be several identical devices in a system, the programmable part of the slave address enables the maximum possible number of such devices to be connected to the I²C-bus. The number of programmable address bits of a device depends on the number of pins available. For example, if a device has 4 fixed and 3 programmable address bits, a total of 8 identical devices can be connected to the same bus.

The I²C-bus committee coordinates allocation of I²C addresses. Further information can be obtained fom the Philips representatives listed on the back cover. Two groups of eight addresses (0000XXX and 1111XXX) are reserved for the purposes shown in Table 2. The bit combination 11110XX of the slave address is reserved for 10-bit addressing (see Section 13.0).

Table 2 Definition of bits in the first byte

Slave address	R/$\overline{W}$ bit	Description
0000 000	0	General call address
0000 000	1	START byte
0000 001	X	CBUS address
0000 010	X	Address reserved for different bus format
0000 011	X	Reserved for future purposes
0000 1XX	X	
1111 1XX	X	
1111 0XX	X	10-bit slave addressing

NOTES:

1) No device is allowed to acknowledge at the reception of the START byte.
2) The CBUS address has been reserved to enable the inter-mixing of CBUS compatible and I²C-bus compatible devices in the same system. I²C-bus compatible devices are not allowed to respond on reception of this address.
3) The address reserved for a different bus format is included to enable I²C and other protocols to be mixed. Only I²C-bus compatible devices that can work with such formats and protocols are allowed to respond to this address.

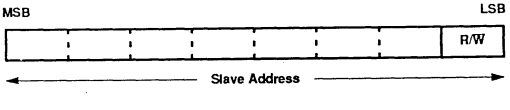


Fig.13 The first byte after the START procedure

I²C-bus specification (including fast-mode)

8.1.1 General call address

The general call address should be used to address every device connected to the I²C-bus. However, if a device doesn't need any of the data supplied within the general call structure, it can ignore this address by not acknowledging. If a device does require data from a general call address, it will acknowledge this address and behave as a slave-receiver. The second and following bytes will be acknowledged by every slave-receiver capable of handling this data. A slave which cannot process one of these bytes must ignore it by not acknowledging. The meaning of the general call address is always specified in the second byte (Fig.14).

There are two cases to consider:

- When the least significant bit B is a 'zero'.
- When the least significant bit B is a 'one'.

When B is a 'zero'; the second byte has the following definition:

- 00000110 (H'06'). Reset and write programmable part of slave address by hardware. On receiving this 2-byte sequence, all devices designed to respond to the general call address will reset and take in the programmable part of their address. Precautions have to be taken to ensure that a device is not pulling down the SDA or SCL line after applying the supply voltage, since these low levels would block the bus
- 00000100 (H'04'). Write programmable part of slave address by hardware. All devices which define the programmable part of their address by hardware (and which respond to the general call address) will latch this programmable part at the reception of this two byte sequence. The device will not reset.
- 00000000 (H'00'). This code is not allowed to be used as the second byte.

Sequences of programming procedure are published in the appropriate device data sheets.

The remaining codes have not been fixed and devices must ignore them.

When B is a 'one'; the 2-byte sequence is a 'hardware general call'. This means that the sequence is transmitted by a hardware master device, such as a keyboard scanner, which cannot be programmed to transmit a desired slave address. Since a hardware master doesn't know in advance to which device the message has to be transferred, it can only generate this hardware general call and its own address - identifying itself to the system (Fig.15).

The seven bits remaining in the second byte contain the address of the hardware master. This address is recognised by an intelligent device, such as a microcontroller, connected to the bus which will then direct the information from the hardware master. If the hardware master can also act as a slave, the slave address is identical to the master address.

In some systems, an alternative could be that the hardware master

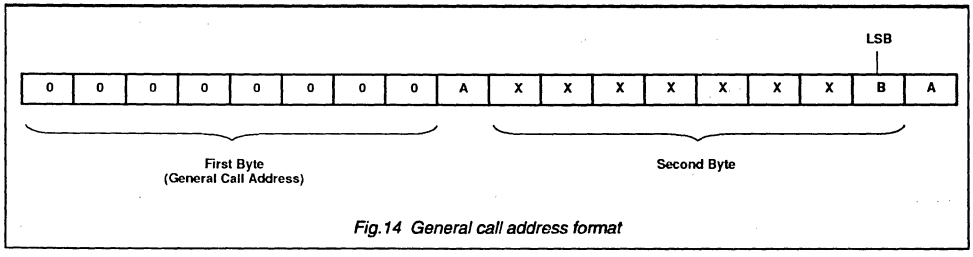


Fig.14 General call address format

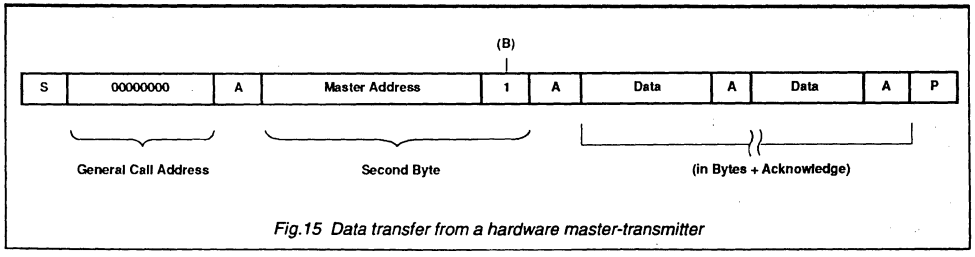


Fig.15 Data transfer from a hardware master-transmitter

I²C-bus specification (including fast-mode)

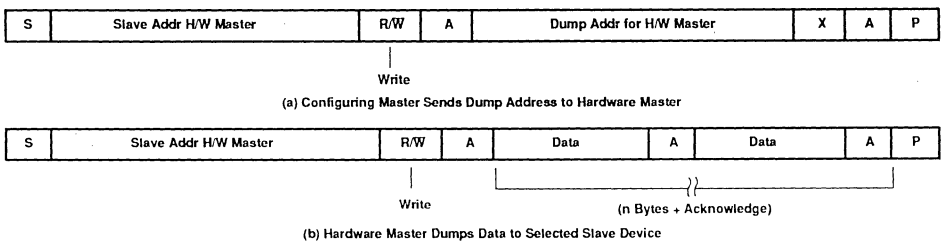


Fig.16 Data transfer by a hardware-transmitter capable of dumping data directly to slave devices (a) Configuring master sends dump address to hardware master (b) Hardware master dumps data to selected slave

transmitter is set in the slave-receiver mode after the system reset. In this way, a system configuring master can tell the hardware master-transmitter (which is now in slave-receiver mode) to which address data must be sent (Fig.16). After this programming procedure, the hardware master remains in the master-transmitter mode.

8.1.2 START byte

Microcontrollers can be connected to the I²C-bus in two ways. A microcontroller with an on-chip hardware I²C-bus interface can be programmed to be only interrupted by requests from the bus. When the device doesn't have such an interface, it must constantly monitor the bus via software. Obviously, the more times the microcontroller monitors, or polls, the bus the less time it can spend carrying out its intended function. There is therefore a speed difference between fast hardware devices and a relatively slow microcontroller which relies on software polling.

In this case, data transfer can be preceded by a start procedure which is much longer than normal (Fig.17). The start procedure consists of:

- A START condition (S)
- A START byte (00000001)
- An acknowledge clock pulse (ACK)
- A repeated START condition (Sr).

After the START condition S has been transmitted by a master which requires bus access, the START byte (00000001) is transmitted. Another microcontroller can therefore sample the SDA line at a low sampling rate until one of the seven zeros in the START byte is detected. After detection of this LOW level on the SDA line, the microcontroller can switch to a higher sampling rate to find the repeated START condition Sr which is then used for synchronization.

A hardware receiver will reset on receipt of the repeated START condition Sr and will therefore ignore the START byte.

An acknowledge-related clock pulse is generated after the START byte. This is present only to conform with the byte handling format used on the bus. No device is allowed to acknowledge the START byte.

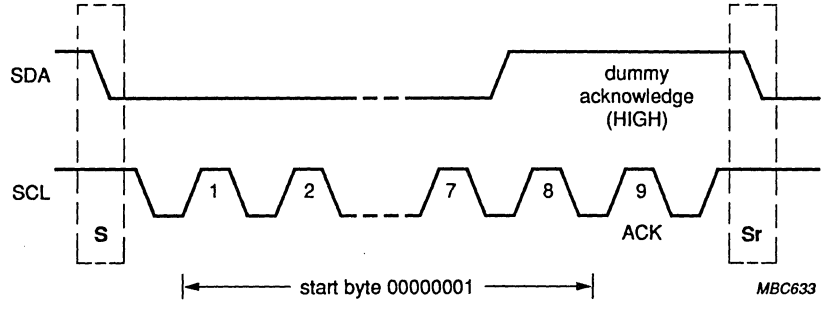


Fig.17 START byte procedure

I²C-bus specification (including fast-mode)

8.1.3 CBUS compatibility

CBUS receivers can be connected to the I²C-bus. However, a third line called DLEN must then be connected and the acknowledge bit omitted. Normally, I²C transmissions are sequences of 8-bit bytes; CBUS compatible devices have different formats.

In a mixed bus structure, I²C-bus devices must not respond to the CBUS message. For this reason, a special CBUS address (0000001X) to which no I²C-bus compatible device will respond, has been reserved. After transmission of the CBUS address, the DLEN line can be made active and a CBUS-format transmission (Fig.18) sent. After the STOP condition, all devices are again ready to accept data.

Master-transmitters can send CBUS formats after sending the CBUS address. The transmission is ended by a STOP condition, recognised by all devices.

NOTE: If the CBUS configuration is known, and expansion with CBUS compatible devices isn't foreseen, the designer is allowed to adapt the hold time to the specific requirements of the device(s) used.

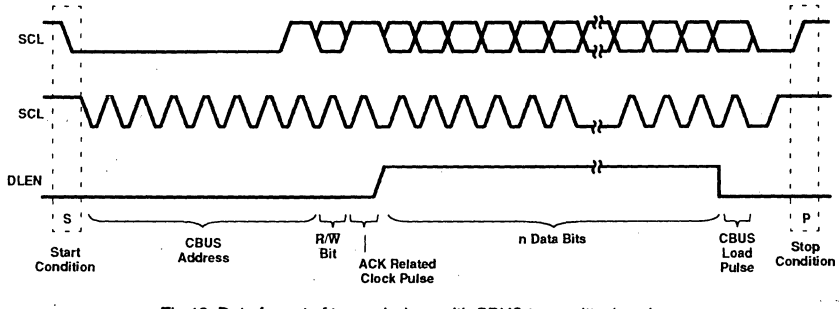


Fig.18 Data format of transmissions with CBUS transmitter/receiver

I²C-bus specification (including fast-mode)

9.0 ELECTRICAL CHARACTERISTICS FOR I²C-BUS DEVICES

The electrical specifications for the I/Os of I²C-bus devices and the characteristics of the bus lines connected to them are given in Tables 3 and 4 in Section 15.

I²C-bus devices with fixed input levels of 1.5 V and 3 V can each have their own appropriate supply voltage. Pull-up resistors must be connected to a 5 V ± 10% supply (Fig.19). I²C-bus devices with input levels related to V_{DD} must have one common supply line to which the pull-up resistor is also connected (Fig.20).

When devices with fixed input levels are mixed with devices with input levels related to V_{DD}, the latter devices must be connected to one common supply line of 5 V ± 10% and must have pull-up resistors connected to their SDA and SCL pins as shown in Fig.21.

Input levels are defined in such a way that:

- The noise margin on the LOW level is 0.1 V_{DD}
- The noise margin on the HIGH level is 0.2 V_{DD}
- Series resistors (R_S) of e.g. 300 Ω can be used for protection against high voltage spikes on the SDA and SCL line due to flash-over of a TV picture tube, for example (Fig.22).

9.1 Maximum and minimum values of resistors R_p and R_s

In a standard-mode I²C-bus system the values of resistors R_p and R_s in Fig.22 depend on the following parameters:

1) Supply voltage
2) Bus capacitance
3) Number of connected devices (input current + leakage current)

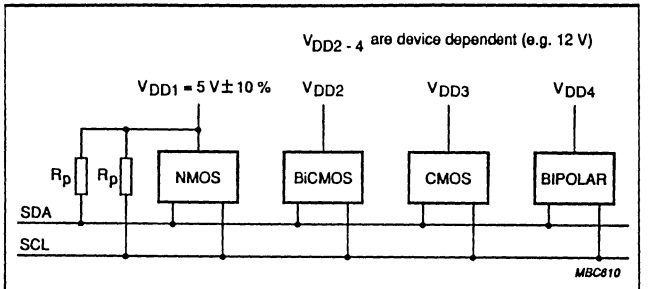


Fig.19 Fixed input level devices connected to the I²C-bus

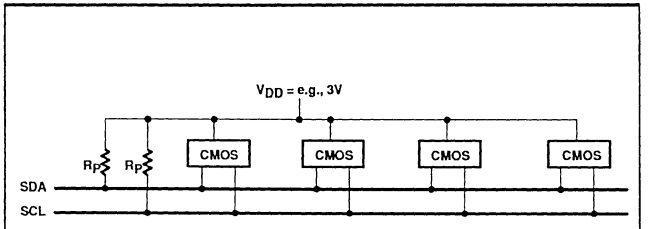


Fig.20 Devices with wide supply voltage range connected to the I²C-bus

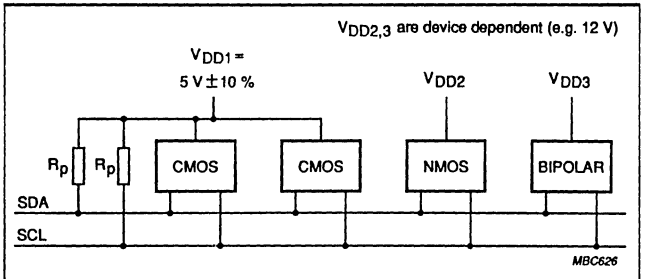


Fig.21 Devices with input levels related to V_{DD} (supply V_{DD1}) mixed with fixed input level devices (supply $V_{DD2, 3}$) on the I²C-bus

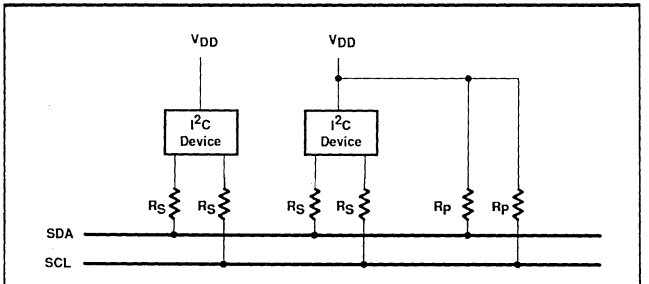


Fig.22 Series resistors (R_s) for protection against high-voltage spikes

I²C-bus specification (including fast-mode)

The supply voltage limits the minimum value of resistor R_p due to the specified minimum sink current of 3 mA at V_{OLmax} = 0.4 V for the output stages. V_{DD} as a function of $R_{p\ min}$ is shown in Fig.23. The desired noise margin of $0.1V_{DD}$ for the LOW level limits the maximum value of R_S. $R_{S\ max}$ as a function of R_p is shown in Fig.24.

The bus capacitance is the total capacitance of wire, connections and pins. This capacitance limits the maximum value of R_p due to the specified rise time. Fig.25 shows $R_{p\ max}$ as a function of bus capacitance.

The maximum HIGH level input current of each input/output connection has a specified maximum value of 10 μA. Due to the desired noise margin of $0.2V_{DD}$ for the HIGH level, this input current limits the maximum value of R_p. This limit depends on V_{DD}. The total HIGH level input current is shown as a function of R_p max in Fig.26.

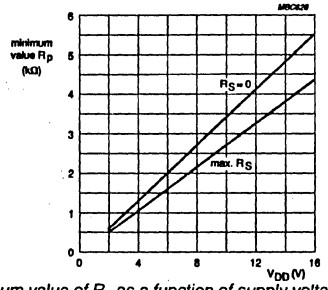


Fig.23 Minimum value of R_p as a function of supply voltage with the value of R_s as a parameter

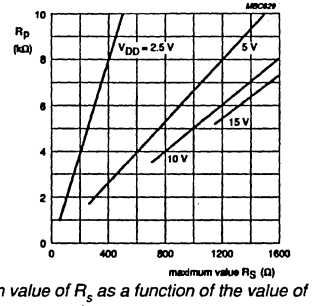


Fig.24 Maximum value of R_s as a function of the value of R_p with supply voltage as a parameter

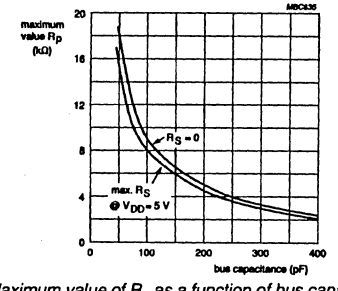


Fig.25 Maximum value of R_p as a function of bus capacitance for a standard-mode I²C-bus

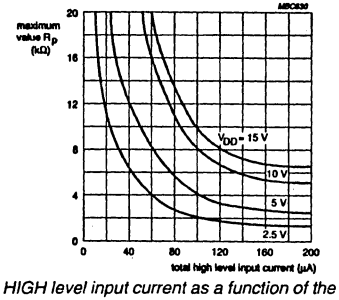


Fig.26 Total HIGH level input current as a function of the maximum value of R_p with supply voltage as a parameter

I²C-bus specification (including fast-mode)

10.0 EXTENSIONS TO THE I²C-BUS SPECIFICATION

The I²C-bus with a data transfer rate of up to 100 kbit/s and 7-bit addressing has now been in existence for more than ten years with an unchanged specification. The concept is accepted world-wide as a de facto standard and hundreds of different types of I²C-bus compatible ICs are available from Philips and other suppliers. The I²C-bus specification is now extended with the following two features:

- A **fast-mode** which allows a fourfold increase of the bit rate to 0 to 400 kbit/s
- **10-bit addressing** which allows the use of up to 1024 additional addresses.

There are two reasons for these extensions to the I²C-bus specification:

- New applications will need to transfer a larger amount of serial data and will therefore demand a higher bit rate than 100 kbit/s. Improved IC manufacturing technology now allows a fourfold speed increase without increasing the manufacturing cost of the interface circuitry
- Most of the 112 addresses available with the 7-bit addressing scheme have been issued more than once. To prevent problems with the allocation of slave addresses for new devices, it is desirable to have more address combinations. About a tenfold increase of the number of available addresses is obtained with the new 10-bit addressing.

All new devices with an I²C-bus interface are provided with the fast-mode. Preferably, they should be able to receive and/or transmit at 400 kbit/s. The minimum requirement is that they can synchronize with a 400 kbit/s transfer; they can then prolong the LOW period of the SCL signal to slow down the transfer. Fast-mode devices must be downward-compatible which means that they must still be able to communicate with 0 to 100 kbit/s devices in a 0 to 100 kbit/s I²C-bus system.

Obviously, devices with 0 to 100 kbit/s I²C-bus interface cannot be incorporated in a fast-mode I²C-bus system because, since they cannot follow the higher transfer rate. Unpredictable states of these devices would occur.

Slave devices with a fast-mode I²C-bus interface can have a 7-bit or 10-bit slave address. However, a 7-bit address is preferred because it is the cheapest solution in hardware and it results in the shortest message length. Devices with 7-bit and 10-bit addresses can be mixed in the same I²C-bus system regardless of whether it is a 0 to 100 kbit/s standard-mode system or a 0 to 400 kbit/s fast-mode system. Existing and future masters can generate 7-bit or 10-bit addresses.

11.0 FAST-MODE

In the fast-mode of the I²C-bus, the protocol, format, logic levels and maximum capacitive load for the SDA and SCL lines given in the previous I²C-bus specification remain unchanged. Changes to the previous I²C-bus specification are:

- The maximum bit rate is increased to 400 kbit/s
- Timing of the serial data (SDA) and serial clock (SCL) signals has been adapted. There is no need for compatibility with other bus systems such as CBUS because they cannot operate at the increased bit rate
- The inputs of fast-mode devices must incorporate spike suppression and a Schmitt trigger at the SDA and SCL inputs
- The output buffers of fast-mode devices must incorporate slope control of the falling edges of the SDA and SCL signals
- If the power supply to a fast-mode device is switched off, the SDA and SCL I/O pins must be floating so that they don't obstruct the bus lines
- The external pull-up devices connected to the bus lines must be adapted to accommodate the shorter maximum permissible rise time for the fast-mode I²C-bus. For bus loads up to 200 pF, the pull-up device for each bus line can be a resistor; for bus loads between 200 pF and 400 pF, the pull-up device can be a current source (3 mA max.) or a switched resistor circuit as shown in Fig.34.

12.0 10-BIT ADDRESSING

The 10-bit addressing does not change the format in the I²C-bus specification. Using 10 bits for addressing exploits the reserved combination 1111XXX for the first 7 bits of the first byte following a START (S) or repeated START (Sr) condition as explained in Section 8.1. The 10-bit addressing does not affect the existing 7-bit addressing. Devices with 7-bit and 10-bit addresses can be connected to the same I²C-bus, and both 7-bit and 10-bit addressing can be used in a standard-mode system (up to 100 kbit/s) or a fast-mode system (up to 400 kbit/s) system.

Although there are eight possible combinations of the reserved address bits 1111XXX, only the four combinations 11110XX are used for 10-bit addressing. The remaining four combinations 11111XX are reserved for future I²C-bus enhancements.

I²C-bus specification (including fast-mode)

12.1 Definition of bits in the first two bytes

The 10-bit slave address is formed from the first two bytes following a START condition (S) or a repeated START condition (Sr).

The first 7 bits of the first byte are the combination 11110XX of which the last two bits (XX) are the two most-significant bits (MSBs) of the 10-bit address; the eighth bit of the first byte is the R/$\overline{W}$ bit that determines the direction of the message. A 'zero' in the least significant position of the first byte means that the master will write information to a selected slave. A 'one' in this position means that the master will read information from the slave.

If R/$\overline{W}$ is 'zero', then the second byte contains the remaining 8 bits (XXXXXXXX) of the 10-bit address. If R/$\overline{W}$ is 'one', then the next byte contains data transmitted from slave to master.

12.2 Formats with 10-bit addresses

Various combinations of read/write formats are possible within a transfer that includes 10-bit addressing. Possible data transfer formats are:

- **Master-transmitter transmits to slave-receiver with a 10-bit slave address. The transfer direction is not changed (Fig.27).** When a 10-bit address follows a START condition, each slave compares the first 7 bits of the first byte of the slave address (11110XX) with its own address and tests if the eighth bit (R/$\overline{W}$ direction bit) is 0. It is possible that more than one device will find a match and generate an acknowledge (A1). All slaves that found a match will compare the 8 bits of the second byte of the slave address (XXXXXXXX) with their own addresses, but only one slave will find a match and generate an acknowledge (A2). The matching slave will remain addressed by the master until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

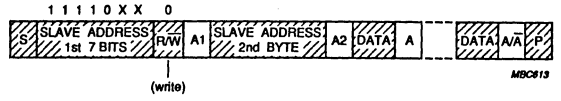


Fig.27 A master-transmitter addresses a slave-receiver with a 10-bit address

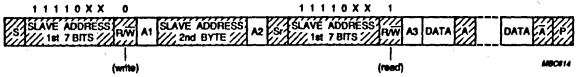


Fig.28 A master-receiver addresses a slave-transmitter with a 10-bit address

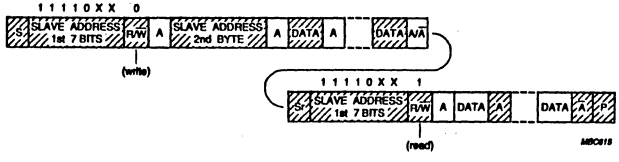


Fig.29 Combined format. A master addresses a slave with a 10-bit address, then transmits data to this slave and reads data from this slave

- **Master-receiver reads slave-transmitter with a 10-bit slave address. The transfer direction is changed after the second R/$\overline{W}$ bit (Fig.28).** Up to and including acknowledge bit A2, the procedure is the same as that described above for a master-transmitter

I²C-bus specification (including fast-mode)

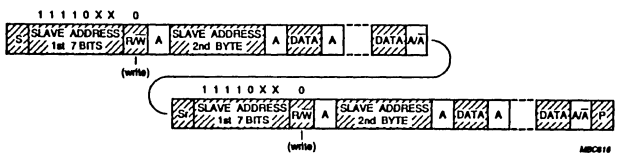


Fig.30 Combined format. A master transmits data to two slaves, both with 10-bit addresses

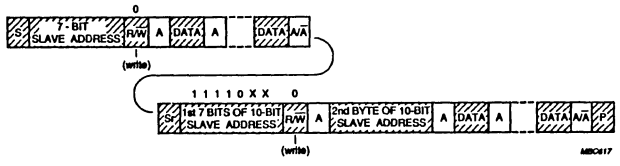


Fig.31 Combined format. A master transmits data to two slaves, one with a 7-bit address, and one with a 10-bit address

addressing a slave-receiver. After the repeated START condition (Sr), a matching slave remembers that it was addressed before. This slave then checks if the first 7 bits of the first byte of the slave address following Sr are the same as before after the START condition (S), and tests if the eighth (R/$\overline{W}$) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a STOP condition (P) or until it receives another repeated START condition (Sr) followed by a different slave address. After Sr, all the other slave devices will also compare the first 7 bits of the first byte of the slave address (11110XX) with their own addresses and test the eighth (R/$\overline{W}$) bit. However, none of them will be addressed because R/$\overline{W}$ = 1 (for 10-bit devices), or the 11110XX slave address (for 7-bit devices) does not match)

- **Combined format. A master transmits data to a slave and then reads data from the same slave (Fig.29).** The same master occupies the bus all the time. The transfer direction is changed after the second R/$\overline{W}$ bit
- **Combined format. A master transmits data to one slave and then transmits data to another slave (Fig.30).** The master occupies the bus all the time
- **Combined format. 10-bit and 7-bit addressing combined in one serial transfer (Fig.31).** After each START condition (S), or each repeated START condition (Sr), a 10-bit or 7-bit slave address can be transmitted. Figure 30 shows how a master-transmits data to a slave with a 7-bit address and then transmits data to a slave with a 10-bit address. The same master occupies the bus all the time.

NOTES:

1) Combined formats can be used, for example, to control a serial memory. During the first data byte, the internal memory location has to be written. After the START condition and slave address is repeated, data can be transferred.
2) All decisions on auto-increment or decrement of previously accessed memory locations etc. are taken by the designer of the device.
3) Each byte is followed by an acknowledgement bit as indicated by the A or $\overline{A}$ blocks in the sequence.
4) I²C-bus compatible devices must reset their bus logic on receipt of a START or repeated START condition such that they all anticipate the sending of a slave address.

13.0 GENERAL CALL ADDRESS AND START BYTE

The 10-bit addressing procedure for the I²C-bus is such that the first two bytes after the START condition (S) usually determine which slave will be selected by the master. The exception is the 'general call' address 00000000 (H'00'). Slave devices with 10-bit addressing will react to a 'general call' in the same way as slave devices with 7-bit addressing (see Section 8.1.1).

Hardware masters can transmit their 10-bit address after a

I²C-bus specification (including fast-mode)

'general call'. In this case, the 'general call' address byte is followed by two successive bytes containing the 10-bit address of the master-transmitter. The format is as shown in Fig. 15 where the first DATA byte contains the eight least-significant bits of the master address.

The START byte 00000001 (H'01') can precede the 10-bit addressing in the same way as for 7-bit addressing (see Section 8.1.2).

14.0 APPLICATION INFORMATION FOR FAST-MODE I²C-BUS DEVICES

14.1 Output stage with slope control

The electrical specifications for the I/Os of I²C-bus devices and the characteristics of the bus lines connected to them are given in Tables 3 and 4 in Section 15.

Figures 32 and 33 show examples of output stages with slope control in CMOS and bipolar technology. The slope of the falling edge is defined by a Miller capacitor (C1) and a resistor (R1). The typical values for C1 and R1 are indicated on the diagrams. The wide tolerance for output fall time t_{OF} given in Table 3 means that the design is not critical. The fall time is only slightly influenced by the external bus load (C_b) and external pull-up resistor (R_p). However, the rise time (t_R) specified in Table 4 is mainly determined by the bus load capacitance and the value of the pull-up resistor.

14.2 Switched pull-up circuit

The supply voltage (V_{DD}) and the maximum output LOW level determine the minimum value of pull-up resistor R_p (see Section 9.1). For example, with a supply voltage of V_{DD} = 5 V ± 10% and $V_{OL\ max.}$ = 0.4 V at 3 mA, $R_{p\ min.}$ = (5.5 − 0.4)/0.003 = 1.7 kΩ. As shown in Fig.35, this value of R_p limits the maximum bus capacitance to about 200 pF to meet the maximum t_R requirement of 300 ns. If the bus has a higher capacitance than this, a switched pull-up circuit as shown in Fig.34 can be used.

The switched pull-up circuit in Fig.34 is for a supply voltage of V_{DD} = 5 V ± 10 % and a maximum capacitive load of 400 pF. Since it is controlled by the bus levels, it needs no additional control signals. During the rising/falling edges, the bilateral switch in the HCT4066 switches pull-up resistor R_p2 on/off at bus levels between 0.8 V and 2.0 V. Combined resistors R_p1 and R_p2 can pull-up the bus line within the maximum specified rise time (t_R) of 300 ns. The maximum sink current for the driving I²C-bus device will not exceed 6 mA at V_{OL2} = 0.6 V, and 3 mA at V_{OL1} = 0.4 V.

Series resistors R_s are optional. They protect the I/O stages of the I²C-bus devices from high-voltage spikes on the bus lines, and minimize crosstalk and undershoot of the bus line signals. The maximum value of R_s is determined by the maximum permitted voltage drop across this resistor when the bus line is switched to the LOW level in order to switch off R_p2.

14.3 Wiring pattern of the bus lines

In general, the wiring must be so chosen that crosstalk and interference to/from the bus lines is minimized. The bus lines are most susceptible to crosstalk and interference at the HIGH level because of the relatively high impedance of the pull-up devices.

If the length of the bus lines on a PCB or ribbon cable exceeds 10 cm and includes the V_{DD} and V_{SS} lines, the wiring pattern must be:

SDA _____
V_{DD} _____
V_{SS} _____
SCL _____

If only the V_{SS} line is included, the wiring pattern must be:

SDA _____
V_{SS} _____
SCL _____

I²C-bus specification (including fast-mode)

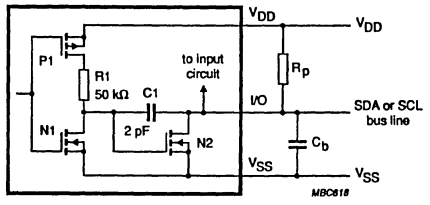


Fig.32 Slope-controlled output stage in CMOS technology

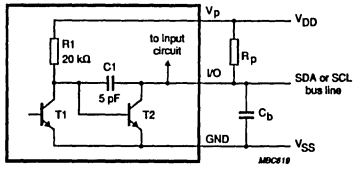
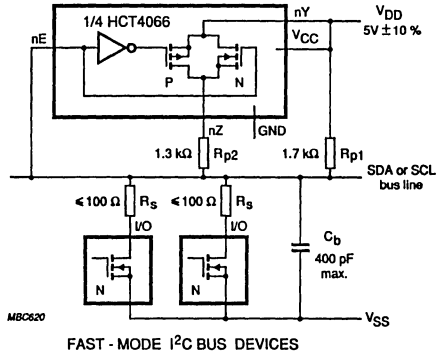


Fig.33 Slope-controlled output stage in bipolar technology



FAST-MODE I²C BUS DEVICES

Fig.34 Switched pull-up circuit

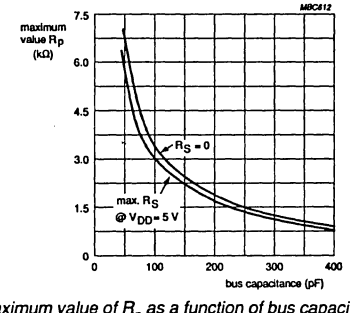


Fig.35 Maximum value of R_p as a function of bus capacitance for meeting the $t_{R\,max.}$ requirement for a fast-mode I^2C-bus

These wiring patterns also result in identical capacitive loads for the SDA and SCL lines. The V_{SS} and V_{DD} lines can be omitted if a PCB with a V_{SS} and/or V_{DD} layer is used.

If the bus lines are twisted-pairs, each bus line must be twisted with a V_{SS} return. Alternatively, the SCL line can be twisted with a V_{SS} return, and the SDA line twisted with a V_{DD} return. In the latter case, capacitors must be used to decouple the V_{DD} line to the V_{SS} line at both ends of the twisted pairs.

If the bus lines are shielded (shield connected to V_{SS}), interference will be minimized. The shielded cable must have low capacitive coupling between the SDA and SCL lines to minimize crosstalk.

14.4 Maximum and minimum values of resistors R_p and R_s

The maximum and minimum values for resistors R_p and R_s connected to a fast-mode I^2C-bus can be determined from Fig.23, 24 and 26 in Section 9.1. Because a fast-mode I^2C-bus has faster rise times (t_R) the maximum value of R_p as a function of bus capacitance is less than that shown in Fig.25 The replacement graph for Fig.25 showing the maximum value of R_p as a function of bus capacitance (C_b) for a fast mode I^2C-bus is given in Fig.35.

I²C-bus specification (including fast-mode)

15.0 ELECTRICAL SPECIFICATIONS AND TIMING FOR I/O STAGES AND BUS LINES

The I/O levels, I/O current, spike suppression, output slope control and pin capacitance for I²C-bus devices are given in Table 3. The I²C-bus timing is given in Table 4. Figure 36 shows the timing definitions for the I²C-bus.

The noise margin for levels on the bus lines for fast-mode devices are the same as those specified in Section 9.0 for standard-mode I²C-bus devices.

The minimum HIGH and LOW periods of the SCL clock specified in Table 4 determine the maximum bit transfer rates of 100 kbit/s for standard-mode devices and 400 kbit/s for fast mode devices. Standard-mode and fast-mode I²C-bus devices must be able to follow transfers at their own maximum bit rates, either by being able to transmit or receive at that speed or by applying the clock synchronization procedure described in Section 6 which will force the master into a wait state and stretch the LOW period of the SCL signal. Of course, in the latter case the bit transfer rate is reduced.

Table 3 Characteristics of the SDA and SCL I/O stages for I²C-bus devices

Parameter	Symbol	standard-mode devices		fast-mode devices		Unit
		Min.	Max.	Min.	Max.	
LOW level input voltage: fixed input levels V_{DD}-related input levels	V_{IL}	 −0.5 −0.5	 1.5 $0.3V_{DD}$	 −0.5 −0.5	 1.5 $0.3V_{DD}$	V
HIGH level input voltage: fixed input levels V_{DD}-related input levels	V_{IH}	 3.0 $0.7V_{DD}$	 *1) *1)	 3.0 $0.7V_{DD}$	 *1) *1)	V
Hysteresis of Schmitt trigger inputs: fixed input levels V_{DD}-related input levels	V_{hys}	 n/a n/a	 n/a n/a	 0.2 $0.05V_{DD}$	 - -	V
Pulse width of spikes which must be suppressed by the input filter	t_{SP}	n/a	n/a	0	50	ns
LOW level output voltage (open drain or open collector): at 3 mA sink current at 6 mA sink current	 V_{OL1} V_{OL2}	 0 n/a	 0.4 n/a	 0 0	 0.4 0.6	V
Output fall time from $V_{IH\,min.}$ to $V_{IL\,max.}$ with a bus capacitance from 10 pF to 400 pF: with up to 3 mA sink current at V_{OL1} with up to 6 mA sink current at V_{OL2}	t_{OF}	 - n/a	 250[2)] n/a	 $20 + 0.1C_b$[2)] $20 + 0.1C_b$[2)]	 250 250[3)]	ns
Input current each I/O pin with an input voltage between 0.4 V and $0.9V_{DD\,max.}$	I_i	−10	10	−10[3)]	10[3)]	µA
Capacitance for each I/O pin	C_i	-	10	-	10	pF

n/a = not applicable

1) maximum $V_{IH} = V_{DD\,max.} + 0.5$ V

2) C_b = capacitance of one bus line in pF. Note that the maximum t_F for the SDA and SCL bus lines quoted in Table 4 (300 ns) is longer than the specified maximum t_{OF} for the output stages (250 ns). This allows series protection resistors (R_s)to be connected between the SDA/SCL pins and the SDA/SCL bus lines as shown in Fig.34 without exceeding the maximum specified t_F.

3) I/O pins of fast-mode devices must not obstruct the SDA and SCL lines if V_{DD} is switched off.

I²C-bus specification (including fast-mode)

Table 4 Characteristics of the SDA and SCL bus lines for I²C-bus devices

Parameter	Symbol	Standard-mode I²C-bus		Fast-mode I²C-bus		Unit
		Min.	Max.	Min.	Max.	
SCL clock frequency	f_{SCL}	0	100	0	400	kHz
Bus free time between a STOP and START condition	t_{BUF}	4.7	-	1.3	-	µs
Hold time (repeated) START condition. After this period, the first clock pulse is generated	$t_{HD;STA}$	4.0	-	0.6	-	µs
LOW period of the SCL clock	t_{LOW}	4.7	-	1.3	-	µs
HIGH period of the SCL clock	t_{HIGH}	4.0	-	0.6	-	µs
Set-up time for a repeated START condition	$t_{SU;STA}$	4.7	-	0.6	-	µs
Data hold time: for CBUS compatible masters (see NOTE, Section 8.1.3) for I²C-bus devices	$t_{HD;DAT}$	 5.0 0[1]	 - -	 - 0[1]	 - 0.9[2]	 µs µs
Data set-up time	$t_{SU;DAT}$	250	-	100[3]	-	ns
Rise time of both SDA and SCL signals	t_R	-	1000	20 + 0.1C_b[4]	300	ns
Fall time of both SDA and SCL signals	t_F	-	300	20 + 0.1C_b[4]	300	ns
Set-up time for STOP condition	$t_{SU;STO}$	4.0	-	0.6	-	µs
Capacitive load for each bus line	C_b	-	400	-	400	pF

All values referred to $V_{IH\ min.}$ and $V_{IL\ max.}$ levels (see Table 3).

[1] A device must internally provide a hold time of at least 300 ns for the SDA signal (referred to the $V_{IH\ min.}$ of the SCL signal) in order to bridge the undefined region of the falling edge of SCL.

[2] The maximum $t_{HD;DAT}$ has only to be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal.

[3] A fast-mode I²C-bus device can be used in a standard-mode I²C-bus system, but the requirement $t_{SU;DAT} \geq 250$ ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line $t_{R\ max.} + t_{SU;DAT} = 1000 + 250 = 1250$ ns (according to the standard-mode I²C-bus specification) before the SCL line is released.

[4] C_b = total capacitance of one bus line in pF.

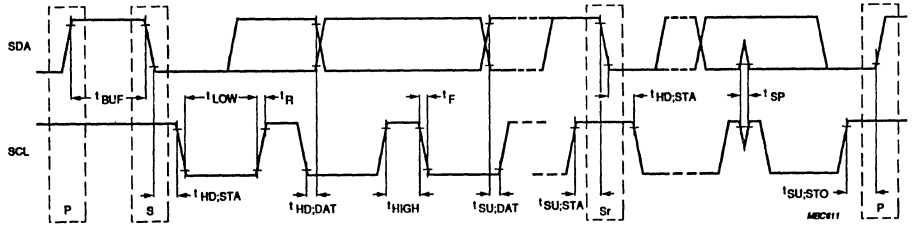


Fig.36 Definition of timing on the I²C-bus

I²C address allocation table

I²C ADDRESS ALLOCATION TABLE

A6–A3	A2–A0							
	0	1	2	3	4	5	6	7
0	General call address	Reserved—	———	———	———	———	———	⟶
1								
2	PCF8200* SAF1135:—	SAA5243* ——— SAA5245*	——— SAA9068!—	⟶ ⟶	SAA9020*	———	———	⟶
3							SAA1136!	
4	SAA1300!— TDA8444!— PCF8574!—	——— ——— ———	——— ——— ———	⟶ ——— ———	 ——— ———	 ——— ———	 ——— ———	 ⟶ ⟶
					PCD3311/A!— PCD3312!—	⟶ ⟶	SA3028	
5								
6								
7	 PCF8576!— PCF8574*— SAA1064!—	 ⟶ ——— ———	PCF8577! ——— ———	PCF8577A! ——— ⟶	PCF8578* PCF8579* ——— ———	⟶ ⟶ ——— ———	PCF8566 !— ———	⟶ ⟶
8	TDA8420!— TDA8421!— TEA6300/T!— TEA6310T!—	⟶ ⟶ TDA8425!	TDA8045*		TDA8422! TDA8461!—	SAA9050* ⟶ SAA9051*	SAA9062* SAA9063* SAA9064*	
9	TDA8440!— PCF8591*—	——— ———	——— ———	——— ———	——— ———	——— ———	——— ———	⟶ ⟶
A	PCF8583*— PCF8570*— PCF8571*— PCF8572*— PCF8582A!—	⟶ ——— ——— ——— ———	 ——— ——— ——— ———	 ——— ——— ——— ———	 ——— ——— ——— ———	 ——— ——— ——— ———	 ——— ——— ——— ———	 ⟶ ⟶ ⟶ ⟶
B	PCF8570*—	———	———	———	———	———	———	⟶
C	TSA5511*— SAB3035*— SAB3036*— SAB3037*— TDA8400!—	——— ——— ——— ——— TEA6000*— TEA6100*— ———	——— ——— ——— ——— TSA6057!— ———	⟶ ⟶ ⟶ ⟶ ⟶ ⟶				
D	TDA8433!— TDA8443A!— TDA8573*—	——— ——— ———	——— ——— ———	——— ——— ⟶	——— ———	——— ———	——— ———	⟶ ⟶
E								
F	Reserved—	———	———	———	———	———	———	⟶

Address Format: | A6 | A5 | A4 | A3 | A2 | A1 | A0 | R/W | Legend: * = R/W, ! = W, : = R

To find a part's address, the most significant 4 bits (A6, A5, A4, and A3) are read from the side of the table. The least significant 3 bits (A2, A1, and A0) are read from the top of the table. For example: SAA1136 is at 36H (0011 110), PCF8577 is at 72H (0111 010). Parts with arrows indicate that a portion of the address is user-configurable. For example: PCF8576 is at address 70H (0111 000) but the last bit (A0) can be set high or low by the user so it can also be at address 71H (0111 001).

I²C bus addresses

ASSIGNED I²C BUS ADDRESSES

PART NUMBER	FUNCTION	I²C ADDRESS						
		A6	A5	A4	A3	A2	A1	A0
—	General call address	0	0	0	0	0	0	0
—	Reserved addresses	0	0	0	0	X	X	X
PCD3311/12	Tone generator DTMF/modem/musical	0	1	0	0	1	0	A
PCF8200	Voice synthesizer (male or female)	0	0	1	0	0	0	0
PCF8566	96-segment LCD driver 1:1–1:4 Mux	0	1	1	1	1	1	A
PCF8568	LCD row driver for dot matrix displays	0	1	1	1	1	0	A
PCF8569	Column driver for dot matrix displays	0	1	1	1	1	0	A
PCF8570/71	256 × 8, 128 × 8 static RAM	1	0	1	0	A	A	A
PCF8570C	256 × 8 static RAM	1	0	1	1	A	A	A
PCF8573	Clock/calendar	1	1	0	1	0	A	A
PCF8574	I²C bus to 8-bit bus converter	0	1	0	0	A	A	A
PCF8574A	I²C bus to 8-bit bus converter	0	1	1	1	A	A	A
PCF6576	160-segment LCD driver 1:1–1:4 Mux	0	1	1	1	0	0	A
PCF8577	64-segment LCD driver 1:1–1:2 Mux	0	1	1	1	0	1	0
PCF8577A	64-segment LCD driver 1:1–1:2 Mux	0	1	1	1	0	1	1
PCF8578	Row/column LCD dot-matrix driver	0	1	1	1	1	0	A
PCF8579	Row/column LCD dot-matrix driver	0	1	1	1	1	0	A
PCF8581	128-byte EEPROM	1	0	1	0	A	A	A
PCF8582	256 × 8 EEPROM	1	0	1	0	A	A	A
PCF8583	256 × 8 RAM with clock/calendar	1	0	1	0	0	0	A
PCF8591	4-channel, 8-bit A/D plus 8-bit D/A	1	0	0	1	A	A	A
PCF8594	512-byte EEPROM	1	0	1	0	A	A	A
SAA1064	4-digit LED driver	0	1	1	1	0	A	A
SAA1136	PCM-Audio indent-word interface	0	0	1	1	1	1	0
SAA1300	5-bit high current driver	0	1	0	0	0	A	A
SAA5243/45	Enhanced teletext circuit	0	0	1	0	0	0	1
SAA7191	S-VHS digital multistandard decoder "square pixel"	1	0	0	0	1	A	1
SAA7192	Digital color space converter	1	1	1	0	0	0	A
SAA7199	Digital encoder	1	0	1	1	0	0	0
SAA9020	Field memory controller	0	0	1	0	1	A	A
SAA9051	Digital multi-standard TV decoder	1	0	0	0	1	0	1
SAA9068	(PIPCO) Picture-in-picture controller	0	0	1	0	0	1	A
SAB3035/36/37	(CITAC) CPU interface for tuning and control	1	1	0	0	0	A	A
SAF1135	Data line decoder	0	0	1	0	0	A	A
TDA4670	Picture signal improvement circuit	1	0	0	0	1	0	0
TDA4680	Video processor	1	0	0	0	1	0	0
TDA8421	Hi-fi stereo audio processor	1	0	0	0	0	0	A
TDA8425	Audio processor w/loudspeaker channel	1	0	0	0	0	0	1
TDA8440	Switch for CTV receivers	1	0	0	1	A	A	A
TDA8442	Interface for color decoders	1	0	0	0	1	0	0
TDA8443	YUV/RGB interface circuit	1	1	0	1	A	A	A
TDA8444	Octuple 6-bit DAC	0	1	0	0	A	A	A
TDA8461	PAL/NTSC color decoder	1	0	0	0	1	0	A
TEA6100	FM/IF and tuning interface	1	1	0	0	0	0	1
TEA6300/6310T	Sound fader control circuit	1	0	0	0	0	0	0
TSA5511/12/14	PLL frequency synthesizer for TV	1	1	0	0	0	A	A
TSA6057	Radio tuning PLL frequency synthesizer	1	1	0	0	0	1	A
UMF1009	Frequency synthesizer	1	1	0	0	0	A	A
—	Reserved addresses	1	1	1	1	X	X	X

X = Don't care.
A = Can be connected high or low by the user.

I²C peripheral selection guide

GENERAL PURPOSE ICs

LCD Drivers

PCF8566	96-segment LCD driver 1:1 – 1:4 Mux
PCF8568	LCD row driver for dot matrix displays
PCF8569	Column driver for dot matrix displays
PCF8576	160-segment LCD driver 1:1 – 1:4 Mux
PCF8577C	64-segment LCD driver 1:1 – 1:2 Mux
PCF8578/79	Row/column LCD dot-matrix driver; 1:8 – 1:32 Mux

I/O Expanders

PCF8574/A	8-bit remote I/O port (I²C-bus to parallel converter)
PCD8584	8-bit parallel to I²C converter
SAA1064	4-digit LED driver
SAA1300	5-bit high-current driver

Data Converters

PCF8591	4-channel, 8-bit Mux ADC + one DAC
TDA8442	Quad 6-bit DAC
TDA8444	Octal 6-bit DAC

Memory

PCF8570/C	256-byte static RAM
PCF8571	128-byte static RAM
PCF8581	128-byte EEPROM
PC.8582	256-byte EEPROM
PCF8583	256-byte RAM/clock/calendar
PCF8594	512-byte EEPROM

Clocks/Calendars

PCF8573	Clock/calendar
PCF8583	Clock/calendar/ 256-byte RAM

68000-Based CMOS Microcontrollers

68070	68000 CPU/MMU/UART/ DMA/timer
93CXXX	UST/I²C/34k ROM/ 512 RAM

80C51-Based CMOS Microcontrollers*

8XCL410	4k ROM/128 RAM, low power
8XC528	32k ROM/512 RAM, T2, WD
8XC552	256-byte RAM/8k ROM/ ADC/UART/PWM
8XC652	256-byte RAM/8k ROM, UART
8XC654	256-byte RAM/16kROM, UART
8XC751	64-byte RAM/2k ROM
8XC752	64-byte RAM/2k ROM, ADC/PWM

8048 Instruction-Set Based CMOS Microcontrollers

PCF84C00	256-byte RAM/bond-out version for prototype development
PCF84C21	64-byte RAM/2k ROM
PCF84C41	128-byte RAM/2k ROM
PCF84C81	256-byte RAM/8k ROM
PCF84C85	256-byte RAM/8k ROM/ Extended I/O
PCF84C430	128-byte RAM/4k ROM/ 96-segment LCD driver

MULTIMEDIA ICs

Video/Radio/Audio

SAA5243	Enhanced Computer Controlled Teletext (ECCT) decoder
SAA5245	Enhanced Computer Controlled Teletext (USECCT) decoder
SAA7191	S-VHS digital multistandard decoder "square pixel"
SAA7192	Digital color space converter
SAA7199	Digital encoder
SAA9041	Digital video teletext (DVTB) processor
SAA9051	7-Bit digital video decoder
SAB3035/36/37	Digital tuning circuits for computer-controlled TV
SAF1135	Dataline 16 decoder for VCR
TDA1551Q	2 x 22W Audio Power Amp
TDA4670	Picture signal improvement circuit
TDA4680	Video processor
TDA6360	5 Band Equalizer
TDA8415/17	Stereo/dual sound processor
TDA8421	Audio processor with a loudspeaker channel and a headphone channel
TDA8425	Audio processor with a loudspeaker channel only
TDA8433	Deflection processor
TDA8440	Video/audio switch
TDA8442	Interface for color decoders
TDA8443/A	YUV/RGB matrix switch
TDA8461	PAL/NTSC color decoder and RGB processor
TDA8466	PAL/NTSC color decoder and RGB processor
TDA9150	Deflection processor
TEA6100	FM/IF and digital tuning IC for computer-controlled radio
TEA6300	Sound fader control and preamplifier/source selector for car radio
TEA6310T	Sound fader control with tone and volume control for car radio
TSA5511/12/14	PLL frequency synthesizer for TV
TSA6057	PLL frequency synthesizer for radio

Telecom

NE5750/51	Audio processor pair
PCD3311/12	Tone generator (DTMF/modem/musical)
PCD3341	Advanced 10 to 110-number repertory dialer with LCD control
PCD3343	Microcontroller with 224-byte RAM/3k ROM
PCD3348	Microcontroller with 256-byte RAM/8k ROM
UMA1000T	Data processor for mobile telephones
UMA1014T	1GHz frequency synthesizer for mobile telephones
UMF1009	Frequency synthesizer

* Also available with extended temperature ranges.

FOR FURTHER INFORMATION ON THESE DEVICES, REFER TO *I²C-PERIPHERALS FOR MICROCONTROLLERS DATA HANDBOOK*, AVAILABLE FROM YOUR LOCAL SIGNETICS SALES OFFICE (SEE SECTION 9 OF THIS BOOK).

Section 3
ACCESS.bus™ Technical Overview

CONTENTS

Notice to prospective developers: ... 159

Introduction ... 159
What is ACCESS.bus? ... 159
ACCESS.bus Hardware ... 160
ACCESS.bus Protocols ... 161

HOW ACCESS.bus Works ... 162
Electrical ... 162
Bus Transactions ... 162
Synchronization ... 162
Byte Framing and Acknowledgement ... 162
Addressing ... 162
Arbitration ... 162
Message Format (Preliminary) ... 164
Control/Status Messages (Preliminary) ... 164
Configuration ... 164
Device Identifiers (Preliminary) ... 165
Device Capabilities Information (Preliminary) ... 165

Application Device Types ... 166
Keyboard Devices ... 166
Locator Devices ... 166
Text Devices ... 166

Timing Rules ... 167
Transaction Timing Rules ... 167
Response Timeouts ... 167

Microcontrollers ... 167

Software Architecture and Development ... 168
Device Firmware Development ... 168
Host Software Architecture ... 168

Development Support ... 169
Digital's TRI/ADD Program ... 169

© by Digital Equipment Corporation

ACCESS.bus™ technical overview

Specifications for this open interconnect are offered by Digital Equipment Corporation and Philips/Signetics

Notice to prospective developers: This document is provided for developers who are interested in learning about a new technology available to the open computing marketplace. The information is subject to change because it is based on specifications that are being completed for availability in the Fall of 1991. In particular, this document contains preliminary protocol information that may not be implemented as described in this version of the Overview. Vendors developing devices based on the ACCESS.bus or integrating it into host systems should check their version of documentation with the TRI/ADD Program (see "Development Support") before beginning any design work.

INTRODUCTION

ACCESS.bus™ (a BUS for connecting ACCESSory devices to a host system) is a peripheral interconnect system defined and developed by Digital Equipment Corporation in partnership with Philips/Signetics and offered to the computer industry as an open standard. This overview aims to introduce the prospective developer of ACCESS.bus systems or peripherals to the essential technical features of this interconnect.

What is ACCESS.bus?

ACCESS.bus is a system for connecting a number of relatively low-speed peripheral devices to a host computer, typically a desktop system, such as a workstation, personal computer, or terminal. The appropriate devices include both interactive peripherals – keyboards, locators, hand held scanners, bar code readers, and magnetic card readers – and non interactive peripherals – printers, plotters, and signal transducers in real-time control applications. Further, the ACCESS.bus protocol is general enough to accommodate a wide range of unusual peripheral types (Figure 1).

ACCESS.bus has the topology of a bus. A single ACCESS.bus on the host can accommodate up to 14 peripheral devices (or more when devices can share a common controller). The total length of the cable connecting the devices on a common ACCESS.bus may be up to eight meters. ACCESS.bus supports a maximum aggregate data throughput of approximately 80 Kbits/sec.

Digital and Philips/Signetics have made the ACCESS.bus an open specification, enabling any vendor to implement it on host systems or in peripheral devices, without fee or royalty. The two companies are working together to promote ACCESS.bus as an industry standard. The Advanced Computing Environment (ACE) initiative has designated ACCESS.bus as an option in the Advanced RISC Computer (ARC) specification. Digital plans to organize a committee of vendors to endorse the ACCESS.bus as a standard and to guide its future development.

Forthcoming Digital workstation products, both host systems and appropriate peripheral devices, will include ACCESS.bus as a built-in feature.

ACCESS.bus offers a number of advantages both to end users and to the developers of systems and peripheral devices. It is an advantage that a host computer needs only one hardware port to connect to a number of devices. The commonality in communication methods for a number of device types leads to economy in software and hardware development. As an open industry standard, ACCESS.bus will stimulate development of diverse peripheral devices, each usable with a number of different types of host system. Further advantages of low cost, ease of implementation, and ease of use are consequences of particular features of the ACCESS.bus design presented in this Overview.

The idea of a bus-topology interconnect for desktop interactive peripherals is fairly new in the computer industry. ACCESS.bus incorporates more sophisticated technology and offers higher performance than any other such system. Moreover, it is the first system of this kind to be offered as an open standard.

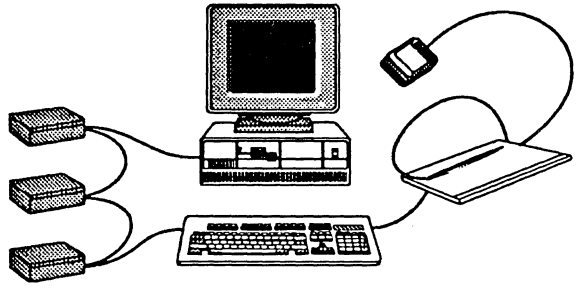


Figure 1. ACCESS bus connects keyboards, locators, and text-type devices to a system.

ACCESS.bus is a trademark of Digital Equipment Corporation.

ACCESS.bus™ technical overview

ACCESS.bus Hardware

At the hardware level, ACCESS.bus is based on the well-established Inter-Integrated Circuit (I^2C) serial bus developed and patented by Philips. The serial bus architecture, in which a single data line caries one bit of information at a time, entails lower costs for cabling, connectors and controller circuitry than parallel bus architectures. As its name indicated, the I^2C serial bus was developed primarily as a means of connecting integrated circuits.

Standard low-cost IC components available from Philips handle the logical complications of the bit-level handshaking. For ACCESS.bus, the relevant components are the microcontrollers of the 80C51 family, which provide the intelligence for executing the ACCESS.bus protocol in peripheral devices and in host systems. More detail on these components are given in the section entitled "MICROCONTROLLERS".

Devices are usually identified with their microcontrollers. However, the ACCESS.bus system includes provisions for controlling up to four subdevices with a single microcontroller. The subdevices may be of diverse functional types. The term "Interface Part" refers to the unique controller of a device, and the term "Application Part" refers to the application-specific functionality. A device has one Interface part and up to four Application Parts (Figure 2).

The physical medium for ACCESS.bus is a shielded cable containing four wires: serial data (SDA), serial clock (SCL), power (+12V), and ground. It uses standard low-cost shielded modular connectors available from AMP and Molex (Figure 3). Shielding of the cables and connectors facilitates making ACCESS.bus-based systems conform to FCC radiation requirements. A typical ACCESS.bus device will have two connectors so that devices may be chained on the single bus; hand-held devices may have a captive cable joined to the bus trunk with a "T" connector.

The way that the serial data and serial clock lines work together to define the information carried on the bus is described in the section entitled "Synchronization". The +12V power line provides up to 500mA to supply the peripheral devices.

The I^2C technology can support clock rates up to 100kHz. The maximum ACCESS.bus data transfer rate of approximately 80 Kbits/sec is derived from the top clock rate by subtracting the overheads imposed by the ACCESS.bus communication protocols for handshaking, addressing, and error control.

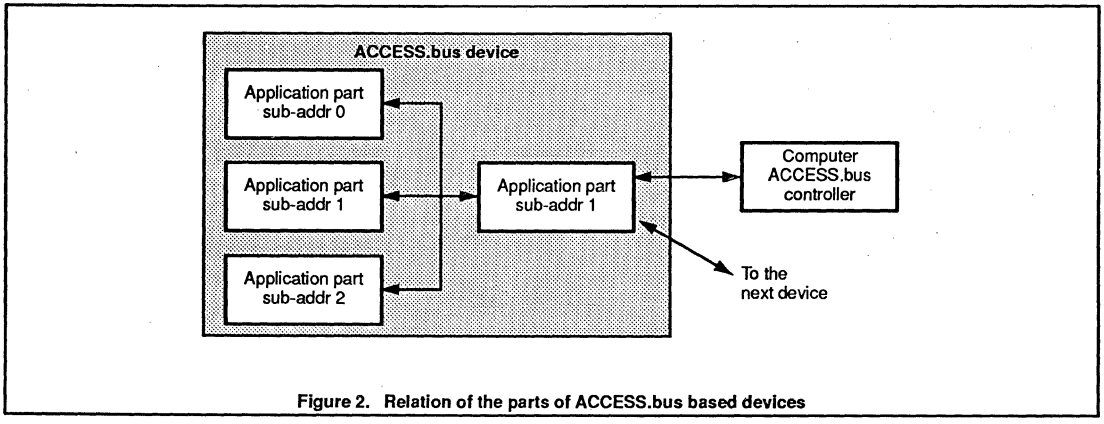


Figure 2. Relation of the parts of ACCESS.bus based devices

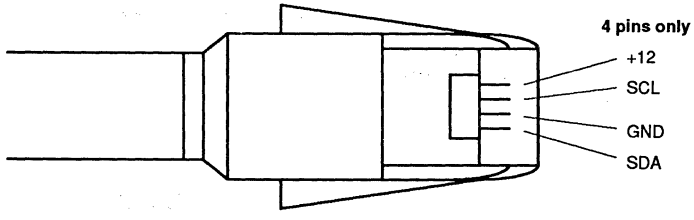


Figure 3. A shielded modular connector with only four pins for connecting the ACCESS.bus cable to a system

ACCESS.bus™ technical overview

ACCESS.bus Protocols

The ACCESS.bus communication protocol is composed of three levels: I²C Protocol, Base Protocol, and Application Protocols.

At the lowest level, nearest the hardware, the basic discipline of the ACCESS.bus is defined as a subset of the Philips Inter-Integrated Circuit (I²C) bus protocol. The simple and efficient I²C Protocol defines a symmetric multi-master bus on which arbitration among contending masters is effected without losing data. I²C provides for cooperative synchronization of the serial clock for exchange of data between bus partners with different maximum clock rates. The I²C Protocol defines a bus transaction scheme with addressing, framing of bits into bytes, and byte-acknowledgement by the receiver. More detail on the I²C Protocol level is given in the section entitled "HOW ACCESS.bus WORKS".

The next ACCESS.bus protocol level is the Base Protocol. This level, common to all types of ACCESS.bus devices, establishes the nature of ACCESS.bus as an asymmetrical interconnect between a host computer and a number of peripheral devices. The host plays a special role as a manager of the ACCESS.bus. Data Communication is always between host and peripheral device and never between two peripherals. While the I²C Protocol provides for mastership by either the sender or the receiver of a bus transaction, in the ACCESS.bus protocol masters are exclusively senders and slaves are exclusively receivers. Of course, the host and all the devices are both master/senders and slave/receivers at different times.

The ACCESS.bus Base Protocol defines the format of an ACCESS.bus message envelope, which is an I²C bus transaction with additional semantics, including checksum reliability control. Further, the Base Protocol defines a set of seven control and status message types which are used in the configuration process.

Two of the unique features of this configuration process are auto-addressing and hot plugging. Auto-addressing refers to the way that devices are assigned unique bus addresses in the configuration process, without the need for setting jumpers or switches on the devices. Hot plugging refers to the ability for attaching or detaching devices while the system is running, without the need for rebooting the host. The means by which the ACCESS.bus protocol provides these feature is discussed in the section entitled "HOW ACCESS.bus WORKS".

The highest level of the ACCESS.bus protocol, the Application Protocol, defines message semantics that are specific to particular functional types of devices. Different device types require different Application Protocols. Application Protocols have been defined so far for three device classes: keyboards, locators, and general text devices. Each of these predefined classes is fairly broad. Thus, the keyboard device protocol provides for communicating essentially all conceivably relevant state information about a keyboard with up to 255 keys. The locator device is defined broadly enough to include not only the most common pointer devices – mice, tablets, trackballs – but also valuator sets (such as dial boxes) with up to 15 valuators and function button boxes with up to 16 buttons. The text device represents essentially any character stream device, such as card readers, scanners, bar code readers, or low speed modems and printers. The predefined Application Protocols are discussed in greater detail in the section entitled "APPLICATION DEVICE TYPES".

A major advantage in designing devices that conform to these general device-type semantics is that they may share device-specific software levels in common, both in the device-resident firmware and in the driver software needed in the host operating system to allow application programs to access the devices.

We anticipate that further device-specific Application Protocols will be defined in the future, as industry standards under the aegis of the planned ACCESS.bus committee. Further, any device vendor may implement a special device protocol within the general message envelope defined by the Base Protocol.

Participation in all three of the protocol levels requires intelligence at the device. The same microcontroller supplies this intelligence at all levels through appropriate firmware. The lower levels of this firmware are likely to be common to many devices. Higher levels of the firmware are expected to be more specific to the device and the application (Figure 4).

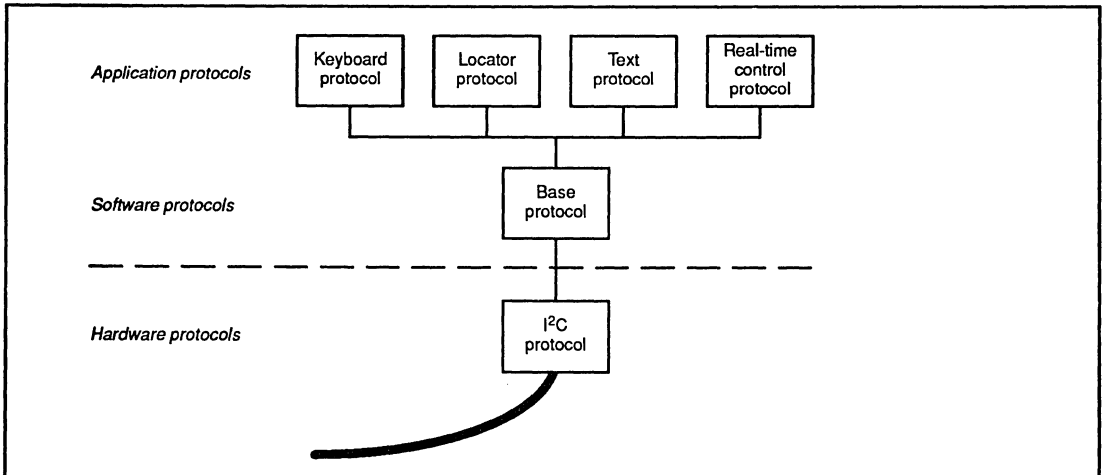


Figure 4. ACCESS.bus protocol hierarchy

ACCESS.bus™ technical overview

HOW ACCESS.bus WORKS

Electrical

The host and devices are connected to both the serial data (SDA) and serial clock (SCL) lines in an "wired-AND" logic configuration. The wired-AND may be implemented by connecting the data and clock output stages of each bus partner to the SDA and SCL lines respectively through open-collector or open-drain transistors. The standard I²C components include such output stages on-chip. The significance of the wired-AND logic is that any attached bus partner may force either of these lines to low (the ground level). When there is no output from any bus partner, the lines are held high by pull-up current sources in the host. And of course, every bus partner can sense the level on both of these lines (Figure 5).

Bus Transactions

During a bus transaction, there is one clock pulse on SCL for each bit transferred on SDA. The SDA information is valid when SCL is high. During a transaction, the SDA must be stable between the rising and falling edges of the SCL pulse; SDA may change state only when SCL is low (Figure 6). SDA transitions when the SCL is high are signals that delimit the bus transaction. When the ACCESS.bus is free, both SCL and SDA are high. A high-to-low SDA transition when SCL is high is a start condition; it signals the beginning of a bus transaction. A bus partner asserts mastership by pulling SDA low when the bus is free. A low-to-high SDA transition when SCL is high is a stop condition; it signals the end of a bus transaction. A master generates a stop condition when it relinquishes mastership (Figure 7).

Synchronization

When a bus partner wishes to assert mastership of a free bus, it generates a start condition by pulling the data SDA low. When SDA is low the new master begins the clock cycle, pulling the SCL low. All bus partners must be able to sense these events, and they respond by pulling all their SCL outputs low and beginning to count off their low periods. When each bus partner has reached the end of its low period, it lets its SCL output go high. Thus the SCL line will remain low for the duration of the longest low clock period among the bus partners.

When all the bus partners have reached the end of their low periods and let their SCL outputs go high, then the SCL goes high. All bus partners must be able to sense this event, and they begin counting their high period. The first bus partner to reach the end of its high period pulls the SCL low again. In this way, all the bus partners simultaneously communicating on the bus are synchronized by a clock pulse whose low is as long as the longest of the low periods and whose high is as long as the shortest of the high periods. This synchronization persists until the master relinquishes the bus by generating a stop condition. The cooperative synchronization is a mechanism by which devices with slower clocks can regulate the operating rate of the bus. However, this mechanism, called "clock stretching", is not the normal means of data stream flow control. The ACCESS.bus protocol provides another mechanism for this purpose. (See the section entitled "Text Devices".)

Byte Framing and Acknowledgement

During this synchronized exchange the master/transmitter puts data on the SDA, one bit for each clock pulse. Eight successive bits comprise a byte, the most significant bit going first.

As the new master puts the first byte on the bus, all the other bus partners participate in the synchronization. The first byte of the transaction contains the address of the intended slave/receiver of the transaction. Each non-master can check the address bits as they appear and cease participating in the synchronization as soon as the address bit on SDA fails to match the corresponding bit of its own address.

At the end of the first byte, the master/transmitter lets its data output go high for the next clock pulse and the slave/receiver whose address matches the transmitted address is obliged to acknowledge receipt of the byte by pulling the SDA low for this pulse. This 1-bit Ack continues after each byte of the bus transaction; the master lets its SDA output go high and the receiver must pull the SCL low. Failure of the receiver to acknowledge a byte is an exception condition, which requires the master to terminate the transaction.

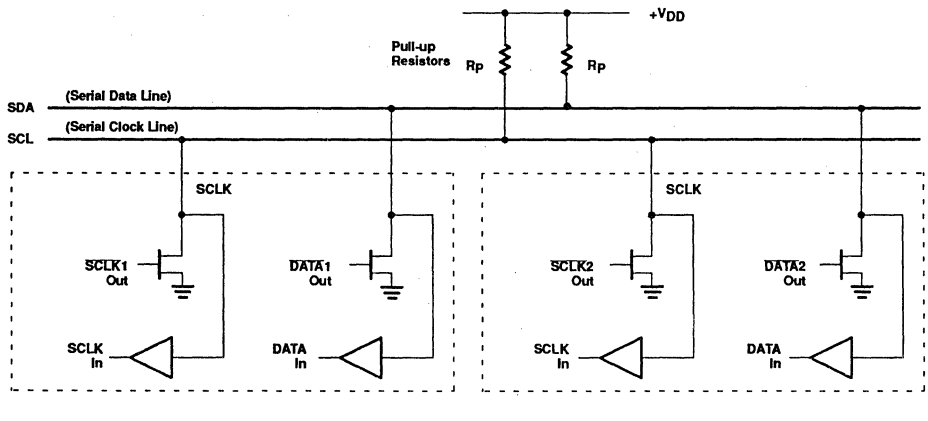


Figure 5. Connection of Devices to the I²C Bus

ACCESS.bus™ technical overview

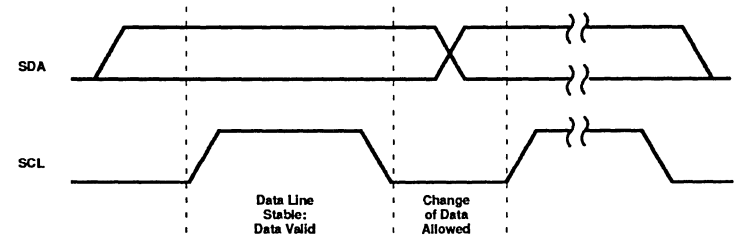


Figure 6. Bit Transfer on the I²C Bus

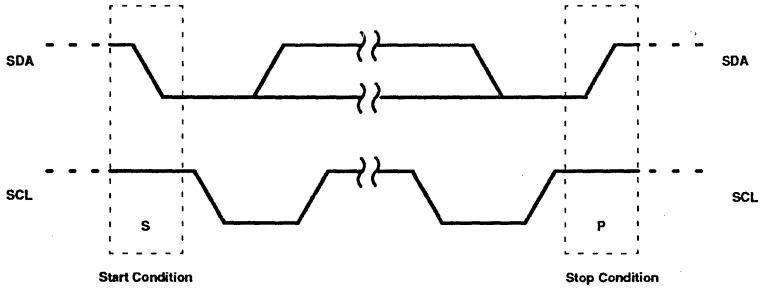


Figure 7. Start and Stop Conditions

Addressing

The slave/receiver of the bus transaction is determined by the address contained in the first byte. I²C uses the seven high-order bits of the first byte for addressing, and bit 0 to indicate whether the master is transmitting or receiving data. In ACCESS.bus, the master is always the transmitter, so bit 0 of the first byte of a transaction is always 0. Of the 128 7-bit addresses, the I²C protocol reserves various ranges for specific kinds of integrated circuit devices. The ACCESS.bus uses only 16 addresses for general microcontrollers. Thus, considered as 8-bit bytes, the valid ACCESS.bus addresses are the even numbers between 50 hexidecimal and 6E hexidecimal, inclusive (80 and 110 decimal, inclusive).

The host computer address is always 50h. In the configuration process to be described below, each peripheral device is assigned a unique address from the set of even numbers between 52h and 6Ch. 6Eh is used as a default address for devices before they have been assigned a unique address.

A device address designates the device controller, or Interface Part, of a peripheral device. When several Application Part subdevices share a controller, they are distinguished by a subaddress field in the third byte of the message, discussed in the section entitled "Message Format (Preliminary)".

Arbitration

What happens when two devices simultaneously assert mastership? While putting data on the SDA, each transmitting master is, of course, independently sensing the state of SDA. Whenever a contending master detects that the state of the SDA is different from the data value it is putting out during a clock high, the contending master backs off, and waits for the stop condition before trying again, Thus, two contending masters will both put data on the bus as long as they are putting out the same data. The first bit where they differ will cause the contender that put out a 1 to back off.

Thus, contending masters trying to send to different bus addresses will resolve the contention by the end of the first byte of the bus transaction. In the ACCESS.bus Base Protocol, the second byte of a transaction is the address of the transmitting master. Thus, as long as bus addresses are unique, the mastership of the bus will be resolved by the end of the second byte of the transaction. However, if two devices have the same address and are trying to send identical messages to a common address, then they will both send the entire message in unison. this situation can happen only during the configuration process before devices have all been assigned unique addresses; it is discussed further in the section entitled "Configuration".

Note that this arbitration mechanism never causes lost data or wasted transmissions, since the addresses of the receiver and transmitter are necessary overhead, in any case, for any sensible bus protocol. Note that bus priorities during arbitration are fixed by the device addresses, first by the address of the receiver, and then, for messages addressed to a common receiver, by the address of the transmitter. Lower addresses have priority over higher addresses. Lockouts of devices with high addresses are prevented by a rule of the Base Protocol that requires partners to wait a minimum time after relinquishing mastership before asserting it again.

ACCESS.bus™ technical overview

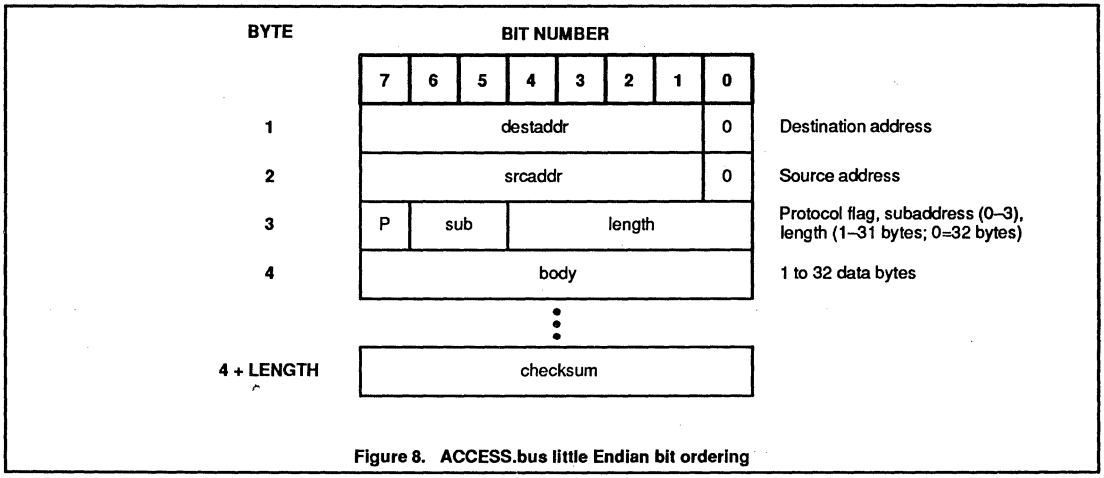


Figure 8. ACCESS.bus little Endian bit ordering

Message Format (Preliminary)

An ACCESS.bus message comprises one I²C bus transaction. It consists of a string of bytes sent by a master/transmitter, each byte acknowledged by a one bit SCL-low Ack from the slave/receiver. The entire transaction is delimited by start and stop conditions generated by the master.

The first byte in the message is the receiver's unique address, as described above in "Addressing". The second byte contains the transmitter's unique address.

The third byte of an ACCESS.bus message comprises three fields. The low order 5 bits provide a byte count for the body of the message, which follows the third byte. A value of zero in this field specifies 32 bytes. Thus, a message body can have 1 to 32 bytes. The message body is followed by a checksum byte, for error control. The checksum is the bitwise XOR of all the preceding bytes of the message.

Bits 5 and 6 of the third byte of the message comprise a subaddress, which can distinguish among up to four Application Part subdevices sharing a common microcontroller, and so a common bus address.

The high order bit of the third byte is a Protocol Flag P to distinguish between data stream messages (P=0) and control/status messages (P=1). The data stream messages carry the application information being exchanges between the device and the host. The control/status messages are used to manage the ACCESS.bus protocol (Figure 8).

Control/Status Messages (Preliminary)

The ACCESS.bus Base Protocol defines a number of control/status messages that pertain to the Interface Parts of devices. These control/status messages are used for the configuration process, in which devices are assigned unique bus addresses and connected with the appropriate drivers in the host. The configuration process is described in detail later. In a control/status message, the message type is indicated by an operation code contained in the first byte of the message body. The various Interface Part control/status messages are as follows:

Computer → Device

Reset()	Force device to power-up state, default address
IdRequest()	Ask device for its identification string
AssignAddress(ID,addr)	Tell device with matching ID its unique bus address
CapRequest(offset)	Ask device to send part of its capabilities

Device → Computer

Attention(status)	Inform computer of device presence and result of power-up reset test
IdReply(ID)	Send device ID string to computer
CapReply(offset,str)	Send part of capability information to computer

The Base Protocol also defines three Application Part control/status messages, applicable to all device types:

AppHardSiga	Provision for a subdevice to generate an interrupt of the host system
AppTest	A message from the host to a peripheral device commanding it to test the Application Part subdevice
AppTestReply	A message by which a device replies to an AppTest command.

In addition, the Application Protocols define further control/status messages that are specific to particular device types. Some of these are discussed in the section entitled "APPLICATION DEVICE TYPES" on the predefined device types.

Configuration

The ACCESS.bus features auto-address and hot-plugging. These features are supported by the ACCESS.bus configuration and process, which uses the seven types of control/status messages. Configuration consists of assigning unique bus addresses to the attached devices and connecting them with the appropriate drivers to provide host-resident application programs with access to the devices.

Configuration occurs when the device is powered-up or when it receives a Reset message. When the system is powered up, so are devices attached to the ACCESS.bus. Otherwise, devices are powered-up when

ACCESS.bus™ technical overview

they are hot-plugged into the bus. A device may have a power source other than the +12V power line of the ACCESS.bus, but is must also be able to sense this line voltage and enter the power-up state when attached to the ACCESS.bus power source. When the system completes its boot up, the host sends a Reset message to all 14 legal device addresses to put all devices in the power-up state.

Usually, when a device is powered up, it performs its specific self-testing. At the conclusion of self-test the device must assume the default address 6Eh and send an Attention message to announce its presence to the host. This message contains a single status byte to inform the host of the results of the power-up self-testing; zero indicates normal results and non-zero values indicate exception conditions that are specific to the device type.

On receiving an Attention message, the host sends an IDRequest message to the default address. Each device at this address replies with an IDReply message containing a unique 28-byte ID string described in the next section. The host is then able to assign unique ACCESS.bus addresses to each of the devices at the default address, by sending AssignAddress messages to the default address. Each AssignAddress message contains in its body the assigned address and the unique ID string of the corresponding device.

Device Identifiers (Preliminary)

During configuration, before address assignment, each device can be identified by a 28-byte unique ID string, which may be partly or entirely encoded in the device ROM. The first 24 bytes of the ID string are understood as ASCII-encoded information characterizing the device type:

- Model revision designation (8 bytes), e.g., "REV X0.2"
- Vendor name (8 bytes), e.g., "DEC"
- Module name (8 bytes), e.g., "LK401".

The first 24 bytes characterize the device's firmware and are encoded in the device ROM.

The remaining 4 bytes of the device ID string are understood as a 32-bit two's complement integer that uniquely identifies the device among devices of the same type. This integer may be provided as a unique serial number contained in the device ROM. Or, in the absence of such a serial number, interactive devices may use a random or arbitrarily determined number for this part of the ID string. As an aid to the host software, the Base Protocol specifies that, in the IDReply message, unique serial numbers be sent as positive integers and randomly generated numbers be sent as negative integers, in the two's complement sense.

In the random number case, it is possible that different devices of the same type may come up with the same 32-bit discriminator. In this case, the different devices will be assigned the same bus address, an undesirable situation. The ACCESS.bus specification suggests a guideline to help avoid identical random identifiers: use the number of cycles of the device's own clock between power up and the time the IDRequest message is received. then natural dispersion of the frequencies of these oscillators is likely to provide unique numbers.

The Base Protocol includes a provision to ensure against the unlikely circumstance that different interactive devices of the same type and without unique serial numbers will generate the same random number. Namely, each such device must send a Reset message to its own assigned address. This self-addressed Reset is sent only once between power-ups or external Resets, just before the device sends the first message instigated by a user action. Of course, the transmitting device will ignore the self-addressed Reset, but other devices possibly at the same address will be reset and will go through configuration again. The Base Protocol specifies that a device using a random number in its ID string shall change that random number after receiving a Reset message. In this way, all interactive devices are guaranteed assignment of unique addresses before sending their first data-carrying messages.

If several noninteractive devices of the same type are to be attached to a single ACCESS.bus, then they must have hardwired unique serial numbers.

During normal operation, the host periodically checks the configuration by sending IDRequest messages to inactive devices. The host also sends IDRequest messages to all assigned addresses whenever it receives an Attention message from a device seeking configuration. The purpose of these IDRequest messages is to verify the current state of the ACCESS.bus – what devices are still connected and which devices are no longer present.

Device Capabilities Information (Preliminary)

In order that a device be accessible to application programs running on the host, it must be connected to an appropriate software driver. Establishing this association is the last phase of configuration.

The appropriate driver will depend on the device type. There may be further parameters that characterize the device and which affect the choice of driver, or which at least must be furnished as arguments to the selected driver. Moreover, the application program may also need to be informed of these device parameters. The device Capabilities Information feature of the ACCESS.bus protocol allows a measure of device independence in the selection of drivers and provides for informing the host software of the device characteristics.

Device Capabilities Information is an explicit statement of a device's functional characteristics that are only implicit in the device type designation contained in the ID string, or that may even vary among individual devices of a given type. For example, the Capabilities Information about a keyboard might include the national alphabet used, or the Capabilities Information about a locator might include its resolution or units.

The Capabilities Information for each device is contained in a single human-readable ASCII-encoded text string stored on the device ROM. The ACCESS.bus Base Protocol defines a simple and compact grammar for building the capabilities string. This grammar provides for hierarchical organization of the Capabilities Information and for the inclusion of subdevice-specific information.

The semantics of the Capabilities Information is carried by keywords. The Base Protocol defines some keywords that can apply to all sorts of devices. Then each Application Protocol will define further keywords that are meaningful only for certain types of devices. To date, the ACCESS.bus Application Protocols define semantics for the Capabilities Information for generic keyboards, locators, and general text devices. The grammar allows for easy extension of the Capabilities Information specification.

ACCESS.bus™ technical overview

An example of a simple mouse Capabilities string might be as follows:

```
(
protocol(locator)
type(mouse)
usage(main)
buttons( 1(L) 2(M) 4(R) )
dim(2)
rel
res(200 inch)
range(-127 127)
d0(dname(X))
d1(dname(Y))
)
```

This would specify that the device is a mouse using the standard locator protocol; that it is a primary device to be used during system startup [usage(main)]; that is has three buttons designated left, middle and right coded with the respective mask values 1, 2, and 4; that it has two degrees of freedom, designated "X" and "Y", using inches as units with 200 counts per inch; that it reports relative values (displacement since last report) in the range -127 to 127 counts.

After assigning a unique address to a device, the host sends it a CapRequest message to command it to send its Capabilities Information in a CapReply message. Of course, the device Capabilities string may well exceed the 31-byte capacity of the message body of a control/status message, so several exchanges of CapRequest/CapReply messages are needed. The two-byte 'offset' argument of each CapRequest specifies where in the Capabilities Information string the fragment should start. This 'offset' value is repeated n the CapReply message, to be used as a check by the host in reassembling the Capabilities string. Thus each CapReply message can contain up to 29 characters of Capabilities Information.

APPLICATION DEVICE TYPES

The initial ACCESS.bus specification defines Application Protocols for three kinds of devices: keyboards, locators, and text devices. An important advantage of developing devices that conform to these defined protocols is the availability of pre-existing software to implement them, in particular, the drivers in the operating software of the host system through which application programs gain access to the devices.

Keyboard Devices

The keyboard device protocol is designed to implement the full functionality of a user interface style like that of the standard Digital LK501 keyboard, while minimizing the amount of state information that must be modeled in both the host and the keyboard, and minimizing also the memory required in the keyboard.

The protocol allows for up to 255 keys. The Capabilities Information semantics provide for specifying the existence of special keypads — numeric, arrows, editing, function keys — the national alphabet, and the parameters of feedback features such as key clicks, bells, and LEDs.

There is just one device data stream message, a keyboard state report, which the keyboard sends on every key transition (key up or key down) and which reports the full state of the keyboard, as a list of all the keys that are down (assumed to be no more than ten).

Device-specific control/status messages from the host include an AppPoll poll command asking the keyboard to report its present state, and commands to produce clicks, bells, and to control the LEDs.

More detailed information can be found in the *ACCESS.bus Keyboard Device Protocol Specification.*

Locator Devices

The *ACCESS.bus Locator Device Specification* provides for a device that has up to 15 degrees of freedom (with 16-bit precision) and up to 16 binary keys or buttons. Thus, in addition to such conventional pointer/locator devices as mouse, tablet, trackball. The ACCESS.bus locator protocol is suitable for valuator sets, such as dial boxes, and function key boxes with up to 16 function keys.

The locator capabilities information provides for specifying the number of switches and their designations (for example, "left", "right", "middle", etc.), whether the locator values are relative (like a mouse) or absolute (like a tablet or dial box), the resolution (counts per unit, and units), the dynamic range, and the names of the locator axes (for example, "x", "y", etc.).

The first 2-byte word of the event report message body contains a mask giving the state of the switches; the remaining words contain the value of each of the locator axes, either the absolute values or the change since the previous report in the case of relative devices. The locator report message is sent either on a regular sampling interval or on receipt of an AppPoll message from the host.

The locator-specific Application Protocol control/status messages are from the host to the device:

AppPoll	Requests the device to report its state
AppSamplingInterval	Sets the device sampling interval or instructs the device to report only when polled

More detailed information can be found in the *ACCESS.bus Locator Device Protocol Specification.*

Text Devices

The ACCESS.bus text device protocol is a simple model for handling devices that transmit and/or receive messages consisting of strings of characters in some fixed character set. The text device data stream message body simply contains 1 to 32 bytes of text, and has no other assumed structure. The text device protocol also includes a provision for high-level flow control.

The defined capabilities information includes the device type (for example, "barcode", "magcode", "smartcard", "port", "printer"), whether the device is for input, output, or both, character set ("ASCII" or "ISO Latin-1"), and the flow control ("input" or "output").

Flow control provides for the case that a data stream receiver cannot handle data as fast as the transmitter can send it, as when a printer with limited buffer capacity receives voluminous data from the host. Flow control uses two text-device-specific Application Protocol control/status messages from the data stream receiver to the data stream sender:

AppPoll	Tells the sender to hold transmission of data stream messages until requested to resume
AppResume(count)	Tells the sender it may resume transmitting up to 'count' characters; if 'count' is zero, then the sender may continue transmitting until receiving an AppHold.

More detailed information can be found in the *ACCESS.bus Text Device Protocol Specification.*

ACCESS.bus™ technical overview

TIMING RULES

To ensure good interactive response and to ensure that all devices will have access to the bus, the ACCESS.bus specifies some rules on transaction timing. Further, specific timeouts are needed to avoid hanging up the interconnect when devices fail or are removed.

Transaction Timing Rules

The ACCESS.bus is designed primarily for interactive devices. A basic objective of the definition of the ACCESS.bus specification is that every interactive device should be able to update the host on its state at least once in every display video frame time. To help meed this criterion, the Base Protocol imposes some rules on the timing of a device's interaction with the bus.

Response Timeouts

In order that a dead or unplugged device will not hang up the system indefinitely, there must be time limits for responding to commands that require a response. The ACCESS.bus protocol specifies that devices shall complete the Reset command within 250ms. Further, a device shall respond to any command requiring a response within 40ms, or, in the case of commands that can be answered by several devices, within 40ms after the last device to respond.

MICROCONTROLLERS

Participation in the several levels of the ACCESS.bus protocol requires intelligence at the devices as well as at the host. Philips produces a number of 8-bit CMOS microcontrollers, of the 80C51 architecture family, featuring some level of I²C interface support on-chip. The use of these low-cost, low pin count, low power-draw components greatly facilitates the design of ACCESS.bus devices. The following are Philips part numbers for the components discussed in this section:

8XCL410
8XC524
8XC528
8XC552
8XC652
8XC654
8XC751
8XC752

Where X is 0, 3, or 7.

Each of these microcontrollers include an 8-bit CPU with a clock oscillator, interrupt structure, and a number of I/O lines. All come in versions with some ROM program memory and a small amount of RAM data memory. In most cases the ROM may be either mask programmable (X=3) or EPROM (X=7). Most of these parts also support up to 64K each of external program and data memories, but accessing external memory requires using some I/O lines as address/data lines. Some of these parts have ROMless versions (X=0), and so would have to make use of external data memory.

Each of these parts contains at least one 16-bit timer/counter, needed for participation in the I²C bus synchronization; some contain as many as three timer/counters, plus a watchdog timer.

Some of these part have analog input pins and on-board A/D convertors as well as pulse-width-modulation (PWM) outputs, both usable for communication with the application-specific circuitry of an ACCESS.bus peripheral device.

All of these components offer open-drain output pins that can be connected to the SDA and SCL lines of the I²C bus. In addition they may contain special registers for controlling the I²C interface functions. Some of these components offer only bit-level I²C interface support in hardware. For example, in the 8XC528 the hardware performs the following functions:

- Generates an interrupt on receiving an I²C start condition
- Recognizes a stop condition and indicates bus-busy or bus-free status
- Latches a received serial bit
- Generates serial clock pulses
- Detects bit-level arbitration loss

But some of the components, such as the 8XCL410, offer byte-level support for the I²C protocol, including byte framing with Ack generation and address comparison and detection. The choice of any of these components as the peripheral component considerably simplifies the development of device firmware and also enables better run-time performance.

Table 1 summarizes some of the relevant parameters and features of each of these microcontrollers, by part number. For detailed descriptions of these components, as well as the complete definition of the 8051 architecture, see the *Data Handbook for 80C51 and Derivative Microcontrollers*, available from Philips/Signetics. This manual also contains a complete specification of the I²C protocol, and provides application notes on I²C implementations, including sample source code for some of the necessary firmware.

In addition to these intelligent microcontrollers, Philips offers the PCD8584 I2C bus controller. This is a non-intelligent component that serves to convert between byte-parallel data and the I²C serial data. It may be used to provide a general microprocessor with an I²C interface.

Table 1. I²C Bus Microcontroller Parameters and Features

DEVICE	ROM	RAM	16-BIT TIMERS	I/O LINES	OTHER FEATURES/LIMITATIONS
8XC751	2K	64	1	19	24-pin Skinny DIP, no ROMless
8XC752	2K	64	1	21	5 x 8-bit A/D, 1 PWM, no ROMless
8XCL410	4K	128	2	32	Low voltage (1.8–6 Volts), byte-level I²C, no EPROM
8XC552	8K	256	3 + watchdog	48	8 x 10-bit A/D, 2 PWM, WD, 3rd Timer
8XC652	8K	256	2	32	I²C
8XC654	16K	256	2	32	No ROMless, I²C
8XC524	16K	512	3 + watchdog	32	WD, I²C
8XC528	32K	512	3 + watchdog	32	WD, I²C

ACCESS.bus™ technical overview

SOFTWARE ARCHITECTURE AND DEVELOPMENT

An ACCESS.bus peripheral requires software at both ends of the bus transaction for managing all levels of the peripheral interaction: the I²C interface, the ACCESS.bus Base Protocol, and the ACCESS.bus Application Protocol. Further, the peripheral device requires software to support communication between the device microcontroller and the application-specific I/O transducer circuitry. Finally, the host system operating software must provide interfaces by which application programs can access both the ACCESS.bus devices and the ACCESS.bus itself. An important advantage of the ACCESS.bus approach is that the lower levels of the interaction are common to diverse device types, so they can be supported by the same or similar software modules.

Device Firmware Development

The microcontroller in the device provides the intelligence for managing the device's participation in all the levels of the ACCESS.bus protocol. Use of the components with hardware I²C interface functionality, described in "MICROCONTROLLERS", can simplify the development of the lowest level of this software. Moreover, because they concern only the bus communication methods that are common to all sorts of peripheral devices, the I²C interface and the ACCESS.bus Base Protocol may well be implemented by reusing software previously developed for some of these components. And devices conforming to the semantics of the predefined standard Application Protocols may also benefit from the availability of some off-the-shelf code at the top level.

Of course, each device vendor will have to develop substantial firmware specific to his or her device. Philips and several third party vendors offer a range of tools to support firmware development for the standard components of the 80C51 family. These tools include cross assemblers and cross compilers for C and PL/M, in-circuit emulators with symbolic debugging and real-time trace support, and EPROM programming equipment. Generally, these software and hardware tools are for use with IBM PC-compatibles as the development platforms.

Host Software Architecture

Vendors of host systems supporting ACCESS.bus will have to supply drivers and other kernel modules to provide access to the ACCESS.bus port, both for application program clients and for other system software, such as the interactive I/O handlers of the window system. In this context, an X Window System server is itself a client of the ACCESS.bus services (Figure 9).

At the lower levels, the I²C level and the ACCESS.bus Base Protocol level, the host is much like a peripheral device; its ACCESS.bus port is based on one of the microcontrollers previously discussed, and its low-level software requirements are similar. The low-level host software is responsible for framing and deframing the ACCESS.bus messages coming from and going to the higher levels.

The heart of the ACCESS.bus access will be in an ACCESS.bus Services layer, which manages the multiplexing of the ACCESS.bus interface resource among the various clients making use of it. This level is responsible for the semantics of the ACCESS.bus messages exchanged with the interface level. The software at this level takes care of the host's responsibilities in the ACCESS.bus configuration process. It performs the mapping between bus addresses and the device handles by which devices are known to applications. It caches device Capabilities Information and makes it available to clients.

At the next level, there are Application Protocol drivers for the predefined device types, providing the bridge between the ACCESS.bus Services layer and the window system services by which applications normally access the common interactive devices. This level should also provide a general-purpose interface by which user-process software may access the bus. This interface may be used for debugging purposes, for access to interactive peripherals independent of the window system, and to accommodate unusual device types.

Of course, all these software levels will generally be specific to the host operating system or window system. Digital will be providing timely ACCESS.bus support in each of its operating systems as hardware interfaces and ACCESS.bus devices are introduced as Digital products. It is expected that standard support for ACCESS.bus will be available for the standard ACE operating systems. Therefore, developers of ACCESS.bus peripherals that conform to the defined standard device type specifications can expect that their product will automatically be compatible with any of the target systems, even in the hot-pluggable sense. Developers of unusual ACCESS.bus peripheral devices will have also to develop the appropriate driver software and to deal with the problems of having it installed on customers' computers.

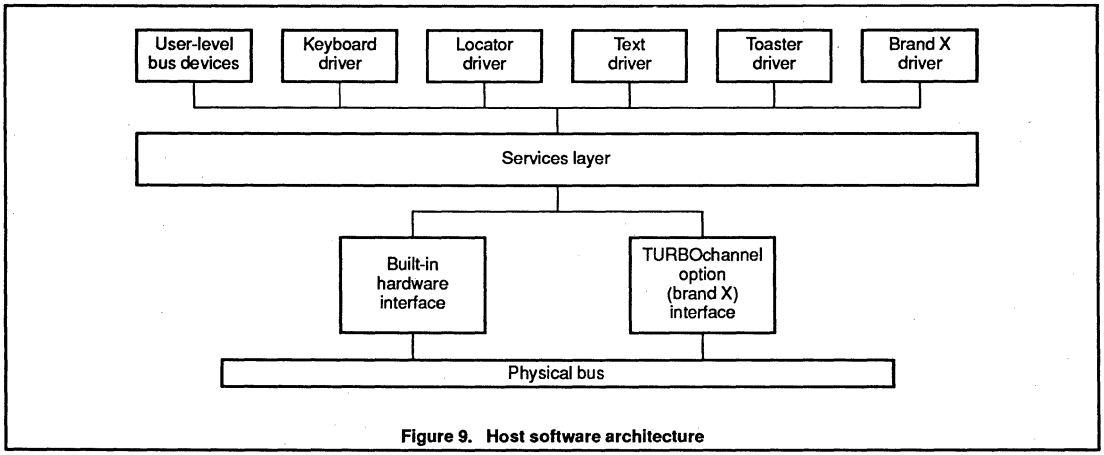


Figure 9. Host software architecture

ACCESS.bus™ technical overview

DEVELOPMENT SUPPORT

Both Digital and Philips/Signetics offer technical support and assistance to developers of ACCESS.bus devices and host systems.

Digital's TRI/ADD Program

Digital's TRI/ADD Program provides support to third party vendors of hardware products compatible with all of Digital's open buses and interconnects, including ACCESS.bus.

Technical services available to TRI/ADD members include free consultation on hardware and software issues, necessary documentation and updates, technical development seminars, and newsletters. The products of TRI/ADD members are listed in Digital's hardware catalogs and can enjoy further marketing support. Further, TRI/ADD products can be supported by Digital's world-wide field service organization. The TRI/ADD Program will also be a source of information on the activities of the planned ACCESS.bus standards committee.

For further information on the TRI/ADD Program and its support of the ACCESS.bus, contact

TRI/ADD Program
Digital Equipment Corporation
100 Hamilton Avenue
Palo Alto, CA 94301

Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specifications defined by Philips.

Section 4
80C51 Family Derivatives

80C51-Based 8-Bit Microcontrollers

CONTENTS

80CL31/80CL51 Overview ... 173
Differences from the 80C51 ... 173
Special Function Registers ... 173
The Interrupt Structure ... 175
Power-Down Mode ... 176
Low Power Consumption ... 176
80CL31/80CL51 Data Sheet ... 177

8XC52 Overview ... 196
Differences from the 80C51 ... 196
Program Memory ... 196
Special Function Registers ... 196
Timer/Counters ... 196
Serial Port ... 197
Timer/Counter 2 Set-up ... 201
Using Timer/Counter 2 to Generate Baud Rates ... 201
Interrupts ... 201
Port Structures ... 201
8032AH/8052AH Data Sheet ... 206
80C32/80C52/87C52 Data Sheet ... 215

8XC053/54 Overview ... 230
The 8XC053/54 On-Screen Display (OSD) Block ... 231
Differences from the 80C51 ... 232
Memory Organization ... 232
Special Function Registers ... 232
Reduced Power Modes ... 232
Interrupts ... 232
83C053/83C054/87C054 Data Sheet ... 234

8XCL410 Overview ... 250
Differences from the 80C51 ... 250
Special Function Registers ... 250
I²C Serial Interface ... 250
The Interrupt Structure ... 253
Power-Down Mode ... 254
Low Power Consumption ... 254
80CL410/83CL410 Data Sheet ... 256

8XC451 Overview ... 277
Differences from the 80C51 ... 277
Special Function Registers ... 277
I/O Port Structure ... 277
Processor Bus Interface ... 277
Standard Quasi-bidirectional I/O Port ... 277
Parallel Printer Port ... 277
80C451/83C451/87C451 Data Sheet ... 280

8XC524/8XC528 Overview ... 297
Differences from the 80C51 ... 297
Data Memory ... 297
Special Function Registers ... 298
Watchdog Timer ... 298
I²C Interface ... 300
Interrupts ... 301
Idle Mode ... 301
Power-Down Mode ... 301
83C524/87C524 Data Sheet ... 302
80C528/83C528/87C528 Data Sheet ... 320

8XC550 Overview ... 340
Differences from the 80C51 ... 340
Special Function Registers ... 340
I/O Port Structure ... 340
A/D Converter ... 340
Sample A/D Routines ... 342
Watchdog Timer ... 342
Interrupts ... 344
Interrupt Control Registers ... 344
Power-Down and Idle Modes ... 344
80C550/83C550/87C550 Data Sheet ... 345

Contents (Continued)

SECTION 4
80C51 FAMILY DERIVATIVES

8XC552/562 Overview ... 363
- Differences from the 80C51 ... 363
  - Program Memory ... 363
  - Data Memory ... 363
  - Special Function Registers ... 363
  - Timer T2 ... 364
  - Timer T3, The Watchdog Timer ... 370
  - Serial I/O ... 371
  - Reset Circuitry ... 405
  - Interrupts ... 405
  - I/O Port Structure ... 409
  - Port 1 Operation ... 409
  - Port 5 Operation ... 409
  - Pulse Width Modulated Outputs ... 409
  - Analog-to-Digital Converter ... 409
  - Power Reduction Modes ... 415
  - Memory Organization ... 417

80C552/83C552/87C552 Data Sheet ... 422

80C562/83C562 Data Sheet ... 444

8XC575 Overview ... 457
- Low Active RESET ... 457
- Timers ... 457
- Programmable Counter Array (PCA) ... 457
  - PCA Capture Mode ... 458
  - 16-bit Software Timer Mode ... 458
  - High Speed Output Mode ... 458
  - Pulse Width Modulator Mode ... 458
- Enhanced UART ... 458
- Analog Comparators ... 458
- Reduced EMI Mode ... 458

80C575/83C575/87C575 Data Sheet ... 461

80C592/83C592/87C592 Data Sheet ... 480

8XC652/654 Overview ... 518
- Differences from the 80C51 ... 518
  - Special Function Registers ... 518
  - I^2C Serial Communication—SIO1 ... 518
  - Idle and Power-Down Operation ... 518
  - ROM Code Protection (83C652/83C654) ... 518
  - Interrupt System ... 520

80C652/83C652/87C652 Data Sheet ... 522

83C654/87C654 Data Sheet ... 541

80CE654/83CE654 Data Sheet ... 560

8XC751 Overview ... 573
- Differences from the 80C51 ... 573
  - Instruction Set ... 573
  - Memory Organization ... 573
  - Special Function Registers ... 573
  - Data Pointer (DPTR) ... 573
  - I/O Port Latches (P0, P1, P3) ... 573
  - I/O Port Structure ... 575
  - Timer/Counter ... 575
  - I^2C Serial Interface ... 575
  - I^2C Interrupts ... 576
  - Interrupts ... 578

83C751/87C751 Data Sheet ... 580

8XC752 Overview ... 591
- Differences from the 80C51 ... 591
  - Instruction Set ... 591
  - Memory Organization ... 591
  - I/O Ports ... 591
  - PWM Outputs ... 591
  - A/D Converter ... 593
  - Counter/Timer ... 594
  - I^2C Serial I/O ... 595
  - Interrupts ... 595
  - Power-Down and Idle Modes ... 595
  - Special Function Registers ... 595
  - Data Pointer ... 595

83C752/87C752 Data Sheet ... 596

8XC851 Overview ... 608
- Differences from the 80C51 ... 608
  - Special Function Registers ... 608

80C851/83C851 Data Sheet ... 610

83C852 Data Sheet ... 624

80CL31/80CL51 OVERVIEW

The 80CL31/80CL51 (hereafter generically referred to as 80CL51) is socket compatible with the 80C51 and has many of the same features, but at a much lower power and clock frequency.

The 80CL51 is a member of the family of low-power 80C51 derivative parts. The features of the 80CL51 include:

- 4k × 8 ROM (80CL51)
- 128 × 8 RAM
- Two 16-bit timer/counters
- A two level nested priority interrupt structure
- Full duplex serial channel
- Thirteen interrupt sources (with eight additional external interrupts)
- Idle and power-down modes
- Interrupt or reset return from power-down
- Six oscillator configurations
  - Quartz crystal
  - Ceramic resonator
  - RC
  - LC
  - External
- Input supply range from 1.8V to 6V
- Operating frequency from DC to 16MHz

The 80CL51 can be operated from a single battery supply which can vary between 1.8V and 6V, and for an AC supply the part requires only a simple voltage regulator. In some cases the part can be operated from an unstabilized supply, eliminating altogether the need for a regulator.

The power consumption of the part is much lower than that of a standard 80C51. Operating at 3.5MHz from a 3V supply, the 80CL51 typically draws less than 1mA of current. The power consumption of the part is directly related to the supply voltage and clock frequency. As the supply voltage or clock frequency are increased, the power consumption also increases. At 12MHz and a supply voltage of 5V, the 80CL51 will draw about 10mA of current, which is slightly less than the current that an 80C51 would require.

The advantage that the 80CL51 has over the 80C51 is in its ability to operate at very low frequencies and supply voltages. This makes the part an ideal choice for applications where power is supplied from batteries, or where low supply voltages or clock frequency are necessary. The 80CL51 features a fully static design. Using the on-chip oscillator, the clock frequency is limited to a minimum of 32kHz, but using an external oscillator the part can be operated down to DC. This means that the clock can be turned off, and when it is started again the microcontroller will continue with the action that it was performing when the clock was stopped. This is something that is impossible with dynamic devices because their internal nodes must be constantly refreshed. The static design of the 80CL51 offers the user the ultimate in power-down modes, because the part can be stopped until it is needed and then started from where it was at when it was stopped, with no loss of internal states or data. The power consumption of the 80CL51 when the clock is stopped is less than 1μA.

Differences from the 80C51

Special Function Registers

The 80CL51 contains most of the special function registers found in the 80C51 as well as four additional SFRs that have been added to handle the eight additional external interrupts.

The interrupt structure on the 80CL51 has been upgraded to include eight additional external interrupts. The IE and IP registers on the 80C51 have had their names changed on the 80CL51 to IEN0 and IP0. In addition, two SFRs have been added to handle the additional external interrupts. The registers are IEN1 and IP1.

Two more SFRs are added to allow the user to set the polarity of the additional external interrupts and to hold the interrupt request flags for those added interrupts. The SFRs are the interrupt polarity register (IX1) and the interrupt request flag register (IRQ1).

Table 1 shows the special function registers, their locations, and their reset values for the 80CL51.

Table 1. 80CL51 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes):										
DPH	High byte	83H									00H
DPL	Low byte	82H									00H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP0*#	Interrupt priority 0	B8H	–	–	–	PS	PT1	PX1	PT0	PX0	xx000000B
			FF	FE	FD	FC	FB	FA	F9	F8	
IP1*#	Interrupt priority 1	F8H	PX9	PX8	PX7	PX6	PX5	PX4	PX3	PX2	00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IEN0*#	Interrupt enable 0	A8H	EA	–	–	ES	ET1	EX1	ET0	EX0	00H
			EF	EE	ED	EC	EB	EA	E9	E8	
IEN1*#	Interrupt enable 1	E8H	EX9	EX8	EX7	EX6	EX5	EX4	EX3	EX2	00H
			C7	C6	C5	C4	C3	C2	C1	C0	
IRQ1*#	Interrupt request flag	C0H	IQ9	IQ8	IQ7	IQ6	IQ5	IQ4	IQ3	IQ2	00H
IX1#	Interrupt polarity	E9H									00H
P0*	Port 0	80H	87	86	85	84	83	82	81	80	FFH
P1*	Port 1	90H	97	96	95	94	93	92	91	90	FFH
P2*	Port 2	A0H	A7	A6	A5	A4	A3	A2	A1	A0	FFH
P3*	Port 3	B0H	B7	B6	B5	B4	B3	B2	B1	B0	FFH
PCON	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
SBUF	Serial data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial control	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer/counter control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
TMOD	Timer/counter mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
TH0	Timer 0 high byte	8CH									00H
TH1	Timer 1 high byte	8DH									00H
TL0	Timer 0 low byte	8AH									00H
TL1	Timer 1 low byte	8BH									00H

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.

The Interrupt Structure

External events and the real-time-driven on-chip peripherals require service by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution, a multiple-source, two-priority level, nested interrupt system is provided. The 80CL51 acknowledges interrupt requests from 13 sources as follows:

Priority	Source	Vector Address	Function
Highest	INT0	0003H	External interrupt 0
	S0	0023H	UART interrupt
	INT5	0053H	External interrupt 5
	T0	000BH	Timer 0 interrupt
	INT6	005BH	External interrupt 6
	INT1	0013H	External interrupt 1
	INT2	003BH	External interrupt 2
	INT7	0063H	External interrupt 7
	T1	001BH	Timer 1 interrupt
	INT3	0043H	External interrupt 3
	INT8	006BH	External interrupt 8
	INT4	004BH	External interrupt 4
Lowest	INT9	0073H	External interrupt 9

There are six special function registers associated with the interrupt portion of the 80CL51. They are IEN0, IP0, IEN1, IP1, IX1, and IRQ1. Following is a detailed description of each.

IEN0—Interrupt Enable register zero. This register has the same function as the IE register found on the 80C51.

IEN0 (A8H)

7	6	5	4	3	2	1	0
EA	–	–	ES	ET1	EX1	ET0	EX0

EA – General Enable/Disable control
0 = All interrupts disabled.
1 = Interrupts can be individually enabled or disabled.
ES – Serial port interrupt enable. (1 = enabled, 0 = disabled)
ET1 – Timer 1 interrupt enable. (1 = enabled, 0 = disabled)
EX1 – External interrupt 1 enable. (1 = enabled, 0 = disabled)
ET0 – Timer 0 interrupt enable. (1 = enabled, 0 = disabled)
EX0 – External interrupt 0 enable. (1 = enabled, 0 = disabled)

IP0—Interrupt Priority register zero. This register has the same function as the IP register on the 80C51.

IP0 (B8H)

7	6	5	4	3	2	1	0
–	–	–	PS	PT1	PX1	PT0	PX0

PS – Serial port interrupt priority. (1 = high, 0 = low)
PT1 – Timer 1 interrupt priority. (1 = high, 0 = low)
PX1 – External interrupt 1 priority. (1 = high, 0 = low)
PT0 – Timer 0 interrupt priority. (1 = high, 0 = low)
PX0 – External interrupt 0 priority. (1 = high, 0 = low)

IEN1—Interrupt Enable register one. This register contains the interrupt enables for the eight external interrupts that have been added to the 80CL51. Clearing (0) the bit in the IEN1 register disables all of the interrupts in this register as well as those in the IEN0 register.

IEN1 (E8H)

7	6	5	4	3	2	1	0
EX9	EX8	EX7	EX6	EX5	EX4	EX3	EX2

EX9-EX2—External interrupt enables for external interrupts 2 – 9. (0 = disabled, 1 = enabled)

IP1—Interrupt priority register one. This register allows the priority level of each of the additional external interrupt enables to be set.

IP1 (D8H)

7	6	5	4	3	2	1	0
PX9	PX8	PX7	PX6	PX5	PX4	PX3	PX2

PX9-PX2—External interrupt priority for external interrupts 2 – 9. (0 = low, 1 = high)

IX1—Interrupt Polarity register. This register allows the programmer to determine the polarity of external interrupts 2 – 9 that will be sensed for the interrupt. If the bit for an interrupt is set to 1, then the interrupt will be triggered by a high input on that external interrupt. If the bit is cleared to 0, then the interrupt will be triggered by a low on that external interrupt.

IX1 (E9H)

7	6	5	4	3	2	1	0
IL9	IL8	IL7	IL6	IL5	IL4	IL3	IL2

IL9-IL2—External Interrupt polarity bits corresponding to external interrupts 9 – 2. (1 = high trigger, 0 = low trigger)

IRQ1—Interrupt Request flag register. This register contains flags that are set when one of the external interrupts 2 – 9 are requested. The flags will only be set if the corresponding interrupt is enabled in the IE1 register. The flags must be cleared by software.

IRQ1 (C0H)

7	6	5	4	3	2	1	0
IQ9	IQ8	IQ7	IQ6	IQ5	IQ4	IQ3	IQ2

IQ9-IQ2—External interrupt request flags for external interrupts 9 – 2.

Power-Down Mode

In addition to being able to reduce the power consumption by stopping the clock, there are both idle and power-down modes available. These operate exactly the same as the idle and power-down modes on the 80C51. There is only one difference and that is that it is possible to terminate a power-down condition with either a reset or an external interrupt.

To be able to wake up the part from the power-down state with an external interrupt, both the PD and IDL bits of the PCON register must be set when entering the power-down mode. If only the PD bit is set, the power-down mode will only be terminated by a hardware reset. With both bits set, an interrupt on any of the additional external interrupts, INT2-INT9, will cause the part to wake up. To ensure that the oscillator is stable before the controller restarts, the internal clock will remain inactive for 1536 oscillator periods after the interrupt is detected. After this, the PD flag will be reset, and the part will be in the idle mode, and the interrupt will be handled in the normal way. Figure 1 shows the different oscillator delays associated with the two methods of waking the part up from the power-down mode.

Low Power Consumption

The 80CL51 is targeted toward low power applications in industrial control, portable instrumentation, intelligent computer peripherals, portable consumer products, and smart cards. Working from a single supply which can vary between 1.8V and 6V, the 80CL51 requires only a simple voltage regulator. In many cases it can be operated from an unstabilized supply, eliminating altogether the need for a regulator. A typical 80CL51 device draws 1mA from a 3V supply when running at a 3.5MHz clock frequency. Current consumption in the idle and power-down modes is reduced further to less than 0.5mA and 1μA, respectively.

The 80CL51 can be operated at clock frequencies up to 16MHz. At these frequencies and a 5V supply voltage, the part will draw current similar to that of a standard 80C51. For the 80CL51, the reduction in power is a function of both the clock frequency and the supply voltage. Power dissipation is reduced by lowering the clock frequency and/or reducing the supply voltage. A standard Philips 80C51 will operate down to a clock frequency of 0.5MHz and a supply voltage of 4V. The advantage of the low-power 80CL51 is that it can be run at frequencies as low as 32kHz with the internal oscillator (DC when an external oscillator circuit is used), and that the voltage supply levels can be reduced down to 1.8V. To obtain maximum power reduction, the part can be operated with both reduced clock frequency and reduced voltage supply.

The low voltage operation of the 80CL51 is due in part to the Philips SACMOS process. This is a self aligned contact CMOS process in which the isolation regions between the contacts and the edge of the isolation have been eliminated. This significantly reduces the size of the die, which in turn reduces the parasitic capacitances and drain resistance. This means that for a given clock speed, parts fabricated in the SACMOS process will require less power, and this is most apparent at low frequencies and voltage supply levels.

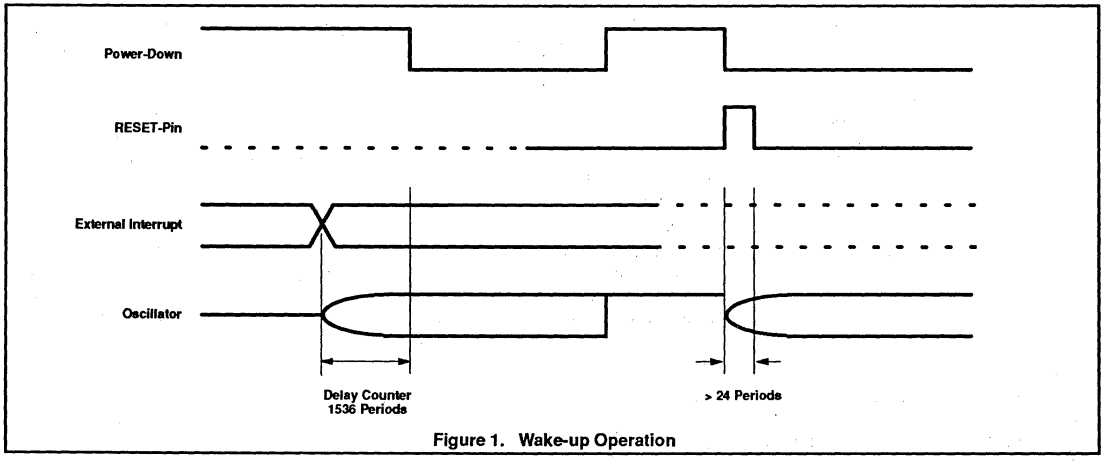


Figure 1. Wake-up Operation

Low voltage/low power single-chip 8-bit microcontroller with UART

80CL31/80CL51

DESCRIPTION

The 80CL31/80CL51 (hereafter generically referred to as 80CL51) is manufactured in an advanced CMOS process that allows the part to operate at supply voltages down to 1.8V and oscillator frequencies down to DC. The 80CL51 has the same instruction set as the 80C51.

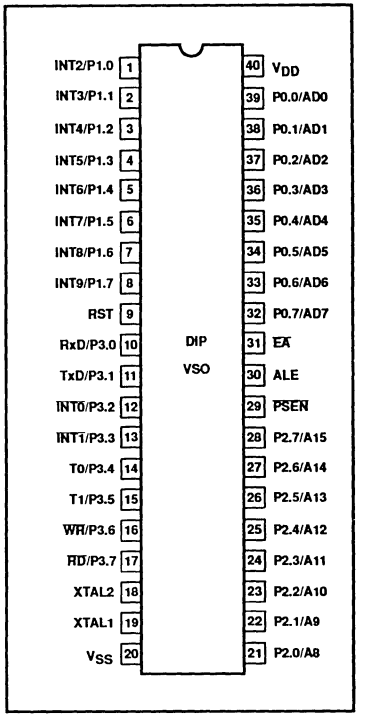
The 80CL51 features a 4k byte ROM (80CL31 is ROMless), 128 bytes RAM (both ROM and RAM are externally expandable to 64k bytes), four 8-bit ports, two 16-bit timer/counters, a serial I/O port for either multi-processor communications, I/O expansion or full duplex UART, a thirteen source, two priority level nested interrupt structure, and on-chip oscillator circuitry suitable for quartz crystal, ceramic resonator, RC, or LC.

The 80CL51 has two reduced power modes that are the same as those on the standard 80C51. The special reduced power feature of this part is that it can be stopped and then restarted. Running from an external clock source, the clock can be stopped and after a period of time restarted. The 80CL51 will resume operation from where it was when the code stopped with no loss of internal state, RAM contents, or Special Function Register contents. If the internal oscillator is used the part cannot be stopped and started, but the power-down mode, which can be terminated via an interrupt, can be used to achieve similar power savings and then restart without loss of on-chip RAM and Special Function Register values.

FEATURES

- Supply voltage from 1.8 to 6.0V
- Operating frequency from 32kHz to 12MHz (see Note 1)
- 80C51 based architecture
  - 4k × 8 ROM (64k external)
  - 128 × 8 RAM (64k external)
  - Four 8-bit I/O ports
  - Two 16-bit timer/counters
  - A thirteen-source, two-level, nested priority interrupt structure
  - 10 external interrupts
- Fully static 80C51 CPU
- Full duplex serial channel
- Two power control modes
  - Idle mode
  - Power-down mode – can be terminated by reset or external interrupt
- Wake-up via external interrupts at port 1
- On-chip oscillator (quartz crystal, ceramic resonator, RC, LC)
- Very low power consumption
- Operating temperature range: –40 to +85°C

PIN CONFIGURATION



NOTE:
1. The currently available product is guaranteed up to 12MHz at 4.5V. A 16MHz device will be made available during 1992.

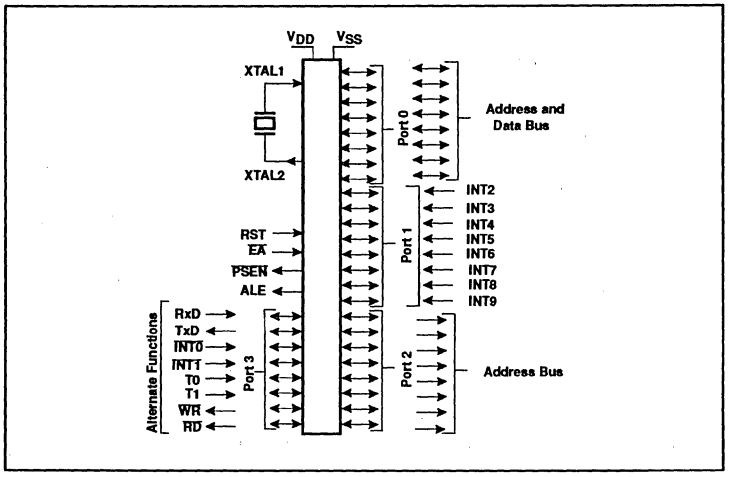
ORDERING CODE

PHILIPS PART ORDER NUMBER PART MARKING		SIGNETICS PART ORDER NUMBER[1]		TEMPERATURE (°C) AND PACKAGE	FREQUENCY
ROMless	ROM	ROMless	ROM		
P80CL31HFP	P80CL51HFP	P80CL31HFP N	P80CL51HFP N	–40 to +85, plastic DIP	32kHZ to 12MHz
P80CL31HFT	P80CL51HFT	P80CL31HFT D	P80CL51HFT D	–40 to +85, plastic VSO	32kHZ to 12MHz

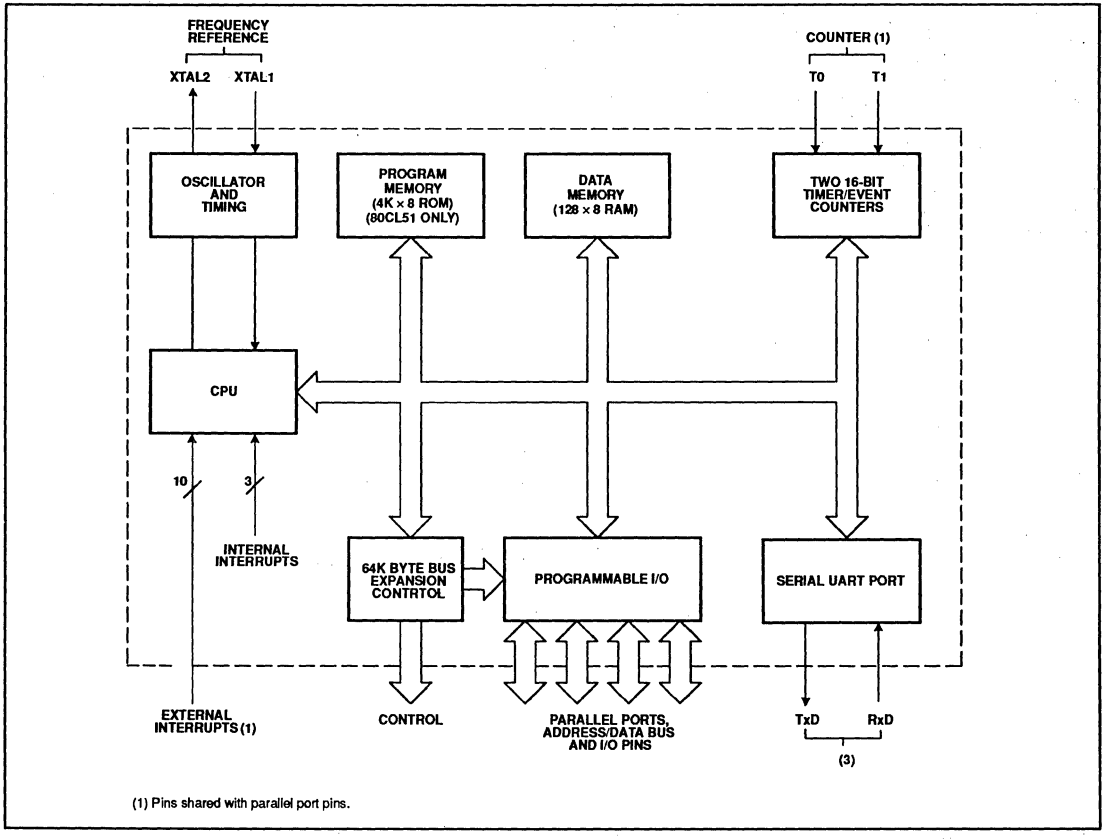
NOTE:
1. Parts ordered by the Signetics part number will be marked with the Philips part marking.

For emulation purposes, the P85CL000 (Piggyback version) with 256 bytes of RAM is recommended.

LOGIC SYMBOL



BLOCK DIAGRAM



(1) Pins shared with parallel port pins.

PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{SS}	20	I	**Ground:** 0V reference.
V_{DD}	40	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s.
P1.0–P1.7	1–8	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Additional functions include:
	1–8	I	**INT2–INT9 (P1.0–P1.7):** Additional external interrupts.
P2.0–P2.7	21–28	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 80C51 family, as listed below:
	10	I	**RxD (P3.0):** Serial input port
	11	O	**TxD (P3.1):** Serial output port
	12	I	**$\overline{INT0}$ (P3.2):** External interrupt 0
	13	I	**$\overline{INT1}$ (P3.3):** External interrupt 1
	14	I	**T0 (P3.4):** Timer 0 external input
	15	I	**T1 (P3.5):** Timer 1 external input
	16	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{DD}.
ALE	30	O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory.
$\overline{PSEN}$	29	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$	31	I	**External Access Enable:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 1FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 0FFFH.
XTAL1	19	I	**Crystal 1:** Input to the inverting oscillator amplifier and input for an external clock source.
XTAL2	18	O	**Crystal 2:** Output from the inverting oscillator amplifier.

PORT OPTIONS

The pins of port 1, port 2, and port 3 may be individually configured with one of the following port options (see Figure 1):

Option 1: **Standard Port**—quasi-bidirectional I/O with pull-up. The strong booster pull-up p1 is turned on for two oscillator periods after a 0-to-1 transition in the port latch. See Figure 1(a).

Option 2: **Open Drain**—quasi-bidirectional I/O with n-channel open drain output. Use as an output requires the connection of an external pull-up resistor. See Figure 1(b).

Option 3: **Push-Pull**—output with drive capability in both polarities. Under this option, pins can only be used as outputs. See Figure 1(c).

The definition of port options for port 0 is slightly different.

Two cases have to be examined. First, accesses to external memory ($\overline{EA}$ = 0 or access above the built-in memory boundary), and second, I/O accesses.

External Memory Accesses

Option 1: True 0 and 1 are written as address to the external memory (strong pull-up is used).

Option 2: An external pull-up resistor is needed for external accesses.

Option 3: True 0 and 1 are written as address to the external memory (strong pull-up is used).

I/O Accesses

Option 1: When writing a 1 to the port latch, the strong pull-up p1 will be on for two oscillator periods. No weak pull-up exists. Without an external pull-up, this option can be used as a high-impedance input.

Option 2: Open drain—quasi-bidirectional I/O with n-channel open drain output. Use as an output requires the connection of an external pull-up resistor. See Figure 1(c).

Option 3: Push-Pull—output with drive capability in both polarities. Under this option, pins can only be used as outputs.

Individual mask selection of the post-reset state is available on any of the above pins. Make your selection by appending "S" or "R" to option 1, 2, or 3 above (e.g., 1S for a standard I/O to be set after RESET or 2R for an open-drain I/O to be reset after RESET.

Option S: **Set**—after reset, this pin will be initialized High.

Option R: **Reset**—after reset, this pin will be initialized Low.

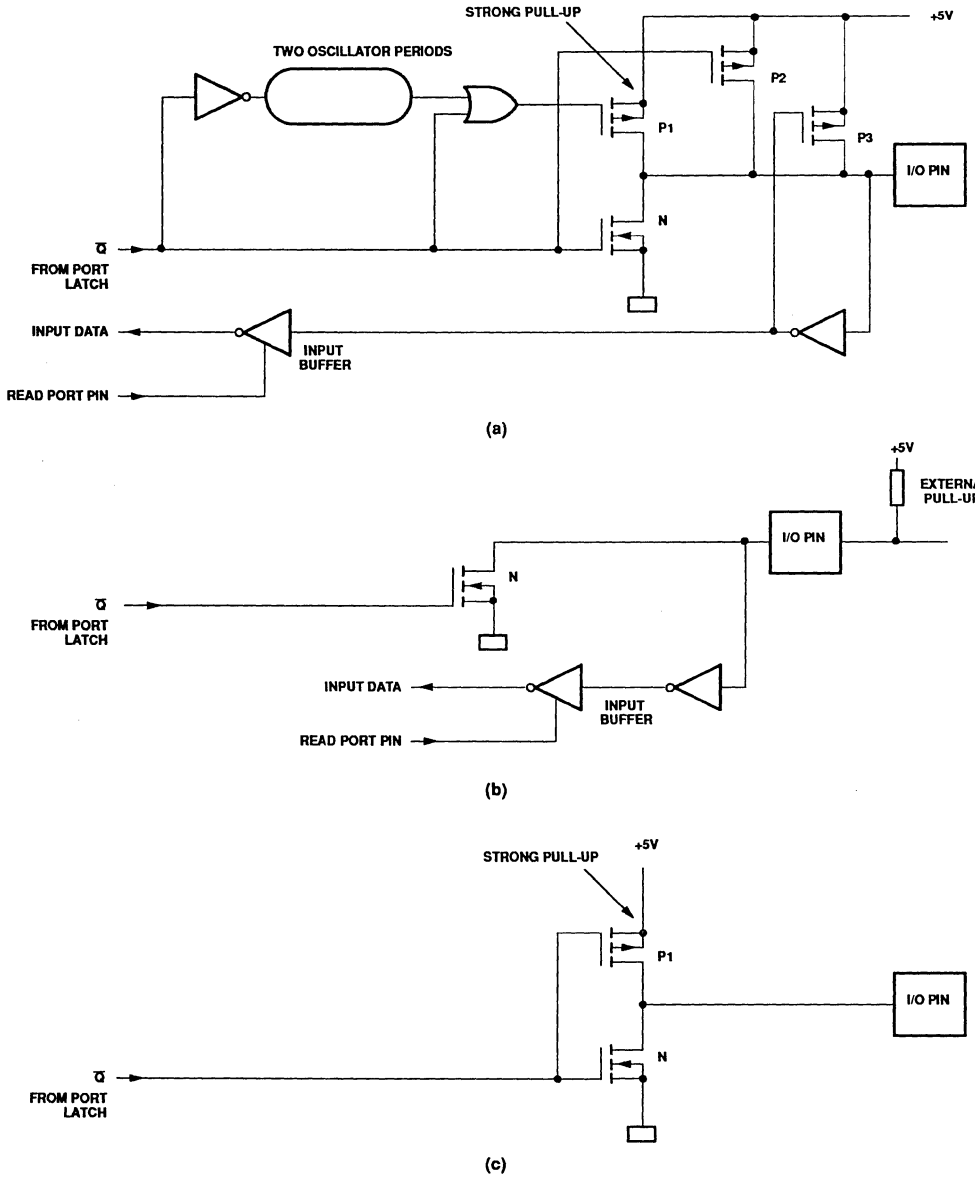


Figure 1. Ports

POWER-DOWN MODE

The instruction setting PCON.1 is the last executed prior to going into the power-down mode. In power-down mode, the oscillator is stopped. The contents of the the on-chip RAM and SFRs are preserved. The port pins output the values held by their respective SFRs. ALE and $\overline{PSEN}$ are held low. Power-down operates in wake-up mode and reset mode.

In the power-down mode, V_{DD} may be reduced to minimize power consumption. However, the supply voltage must not be reduced until the power-down mode is active, and must be restored before the hardware reset is applied and frees the oscillator. Reset must be held active until the oscillator has restarted and stabilized.

Wake-Up Mode

Setting both PD and IDL flags in the PCON register forces the controller into the power-down mode. Setting both flags enable the controller to be woken-up from the power-down mode with either the external interrupts INT2–INT9, or a reset operation.

An external interrupt INT2–INT9 at port 1 releases both the oscillator and the delay counter. To ensure that the oscillator is stable before the controller restarts, the internal clock will remain inactive for 1536 oscillator periods after the interrupt is detected. After this, the PD flag will be reset, the controller is now in the Idle mode and the interrupt will be handled in the normal way.

Reset Mode

Setting only the PD bit in the PCON register again forces the controller into the power-down mode, but in this case it can only be restored to normal operation with a direct reset operation.

IDLE MODE

The instruction that sets PCON.0 is the last instruction executed before going into idle mode. In idle mode, the internal clock is stopped for the CPU, but not for the interrupt, timer, and serial port functions. The CPU status is preserved along with the stack pointer, program counter, program status word and accumulator. The RAM and all other registers maintain their data during idle mode. The port pins retain the logical states they held at idle mode activation. ALE and $\overline{PSEN}$ hold at the logic high level.

There are two methods used to terminate the idle mode. Activation of any interrupt will cause PCON to be cleared by hardware; terminating idle mode. The interrupt is serviced, and following the instruction RETI, the next instruction to be executed will be the one following the instruction that put the device in the the idle mode.

Flag bits GF0 and GF1 can be used to determine whether the interrupt was received during normal execution or idle mode. For example, the instruction that writes to PCON.0 can also set or clear one or both flag bits. When idle mode is terminated by an interrupt, the service routine can examine the status of the flag bits.

The second method of terminating the idle mode is with an external hardware reset. Since the oscillator is still running, the hardware reset is required to be active for only two machine cycles to complete the reset operation. Reset redefines all SFRs, but does not affect the on-chip RAM.

The status of the external pins during idle and power-down mode is shown in Table 1. If the power-down mode is activated while accessing external memory, port data held in the special function register P2 is restored to port 2. If the data is a logic 1, the port pin is held high during the power-down mode.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Floating	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Floating	Data	Data	Data

INTERRUPT SYSTEM

External events and the real-time-driven on-chip peripherals require service by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution, a multiple-source, two-priority level, nested interrupt system is provided. The 80CL51 acknowledges interrupt requests from thirteen sources, as follows:

- INT0 and INT1
- Timer 0 and timer 1
- UART serial I/O interrupt
- INT2 to INT9 (port 1)

Each interrupt vectors to a separate location in program memory for its service routine. Each source can be individually enabled or disabled by corresponding bits in the internal enable registers (IEN0, IEN1) The priority level is selected via the interrupt priority register (IP0, IP1). All enabled sources can be globally disabled or enabled.

External Interrupts INT2–INT9

Port 1 lines serve an alternative purpose as eight additional interrupts INT2–INT9. When enabled, each of these lines can "wake-up" the device from power-down mode. Using the IX1 register, each pin may be initialized to either active high or low. IRQ1 is the interrupt request flag register. Each flag, if the interrupt is enabled, will be set on an interrupt request but it must be cleared by software.

IEN1 (E8H)
Interrupt enable register

7	6	5	4	3	2	1	0
EX9	EX8	EX7	EX6	EX5	EX4	EX3	EX2

Bit	Symbol	Function
IEN1.7	EX9	Enable external interrupt 9
IEN1.6	EX8	Enable external interrupt 8
IEN1.5	EX7	Enable external interrupt 7
IEN1.4	EX6	Enable external interrupt 6
IEN1.3	EX5	Enable external interrupt 5
IEN1.2	EX4	Enable external interrupt 4
IEN1.1	EX3	Enable external interrupt 3
IEN1.0	EX2	Enable external interrupt 2

where 0 = interrupt disabled
1 = interrupt enabled

IP1 (F8H)
Interrupt priority register

7	6	5	4	3	2	1	0
PX9	PX8	PX7	PX6	PX5	PX4	PX3	PX2

Bit	Symbol	Function
IP1.7	PX9	External interrupt 9 priority level
IP1.6	PX8	External interrupt 8 priority level
IP1.5	PX7	External interrupt 7 priority level
IP1.4	PX6	External interrupt 6 priority level
IP1.3	PX5	External interrupt 5 priority level
IP1.2	PX4	External interrupt 4 priority level
IP1.1	PX3	External interrupt 3 priority level
IP1.0	PX2	External interrupt 2 priority level

Interrupt priority is as follows:
0 – low priority
1 – high priority

IX1 (E9H)
Interrupt polarity register

7	6	5	4	3	2	1	0
IL9	IL8	IL7	IL6	IL5	IL4	IL3	IL2

Bit	Symbol	Function
IX1.7	IL9	External interrupt 9 polarity level
IX1.6	IL8	External interrupt 8 polarity level
IX1.5	IL7	External interrupt 7 polarity level
IX1.4	IL6	External interrupt 6 polarity level
IX1.3	IL5	External interrupt 5 polarity level
IX1.2	IL4	External interrupt 4 polarity level
IX1.1	IL3	External interrupt 3 polarity level
IX1.0	IL2	External interrupt 2 polarity level

Writing either a "1" or "0" to an IX1 register bit sets the priority level of the corresponding external interrupt to active High or Low, respectively.

IRQ1 (C0H)
Interrupt request flag register

7	6	5	4	3	2	1	0
IQ9	IQ8	IQ7	IQ6	IQ5	IQ4	IQ3	IQ2

Bit	Symbol	Function
IRQ1.7	IQ9	External interrupt 9 request flag
IRQ1.6	IQ8	External interrupt 8 request flag
IRQ1.5	IQ7	External interrupt 7 request flag
IRQ1.4	IQ6	External interrupt 6 request flag
IRQ1.3	IQ5	External interrupt 5 request flag
IRQ1.2	IQ4	External interrupt 4 request flag
IRQ1.1	IQ3	External interrupt 3 request flag
IRQ1.0	IQ2	External interrupt 2 request flag

Priority	Vector	Source
X0 (highest)	0003H	External 0
S0	0023H	UART
X5	0053H	External 5
T0	000BH	Timer 0
X6	005BH	External 6
X1	0013H	External 1
X2	003BH	External 2
X7	0063H	External 7
T1	001BH	Timer 1
X3	0043H	External 3
X8	006BH	External 8
X4	004BH	External 4
X9 (lowest)	0073H	External 9

Register	Function	SFR Address
IX1	Interrupt polarity register	E9H
IRQ1	Interrupt request flag register	C0H
IEN0	Interrupt enable register	A8H
IEN1	Interrupt enable register (INT2–INT9)	E8H
IP0	Interrupt priority register	B8H
IP1	Interrupt priority register (INT2–INT9)	F8H

OSCILLATOR CIRCUITRY

The on-chip oscillator circuitry of the 80CL51 is a single stage inverting amplifier biased by an internal feedback resistor. (See Figure 2.) The oscillator can be operated with a quartz crystal, ceramic resonator, LC network or RC network. See Figure 3 for different configurations. When ordering parts, it is necessary to specify an oscillator option. The options are: RC when an RC network will be used, OSC 2 for oscillator operation below 4MHz, OSC 3 for oscillator operation from 4MHz to 10MHz, OSC 4 for oscillator operation above 10MHz, and 32kHz if 32kHz to 400kHz operation is desired.

For operation as a standard quartz oscillator, no external components are needed (except at 32KHz). When using external capacitors, ceramic resonators, coils, and RC networks to drive the oscillator, five different configurations are supported (see Figure 3 and Table 2).

In the power-down mode the oscillator is stopped and XTAL1 is pulled high. The oscillator inverter is switched off to ensure no current will flow. To drive the device with an external clock source, apply the external clock signal to XTAL1, and leave XTAL2 to float, as shown in Figure 3(f). There are no requirements on the duty cycle of the external clock, since the input to the internal clocking circuitry is split using a flip-flop.

The following options are provided for optimum on-chip oscillator performance. Please state option when ordering:

32kHz: Figure 3(c). An option for 32kHz clock applications with external trimmer for frequency adjustment.

A 4.7MΩ bias resistor must be connected in parallel with the crystal.

Osc.2: Figure 3(e). An option for low-power, low-frequency operations using LC components or quartz.

Osc.3: An option for medium frequency range applications.

Osc.4: An option for high frequency range applications.

RC: Figure 3(g). An option for an RC oscillator.

The equivalent circuit data of the internal oscillator compares with that of matched crystals.

The externally adjustable RC oscillator has a frequency range from 100kHz to 500kHz. (See Figure 5.)

Power-on Reset

The 80CL51 contains on-chip circuitry which switch the port pins to the customer-defined logic level as soon as V_{DD} exceeds 1.3V. (See Figures 6 and 7.) As soon as the minimum supply voltage is reached, the oscillator will start up. However, to ensure that the oscillator is stable before the controller starts, the clock signals are gated away from the CPU for a further 1536 oscillator periods.

An hysteresis of approximately 100mV at a typical power-on switching level of 1.3V will ensure correct operation.

An automatic reset can be obtained at power-on by connecting the RST pin to V_{DD} via a 10µF capacitor. At power-on, the voltage on the RST pin is equal to V_{DD} minus the capacitor voltage, and decreases from V_{DD} as the capacitor discharges through the internal resistor R_{RST} to ground. The larger the capacitor, the more slowly V_{RST} decreases. V_{RST} must remain above the lower threshold of the Schmitt trigger long enough to effect a complete reset. The time required is the oscillator start-up time, plus 2 machine cycles.

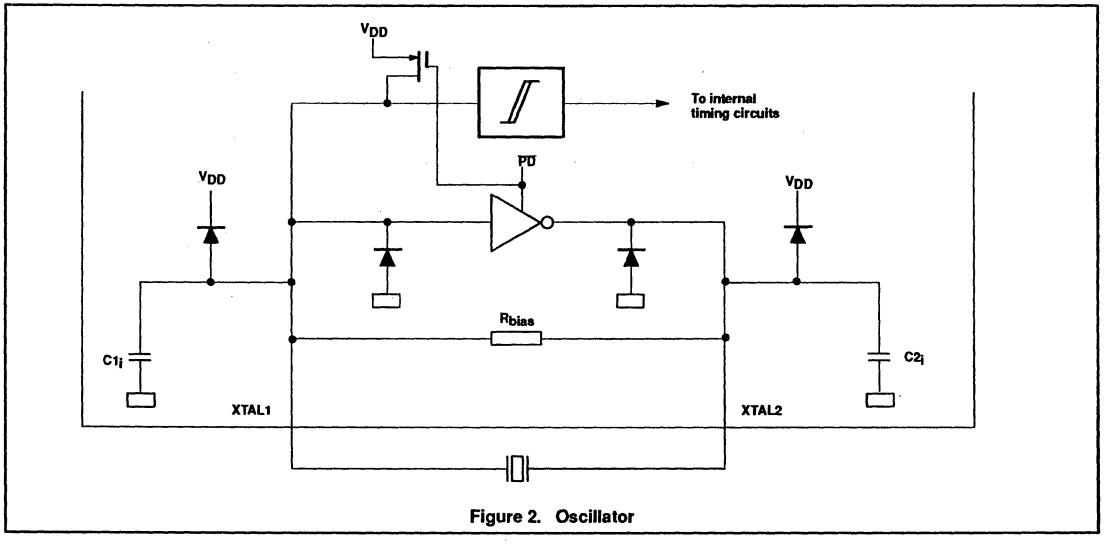


Figure 2. Oscillator

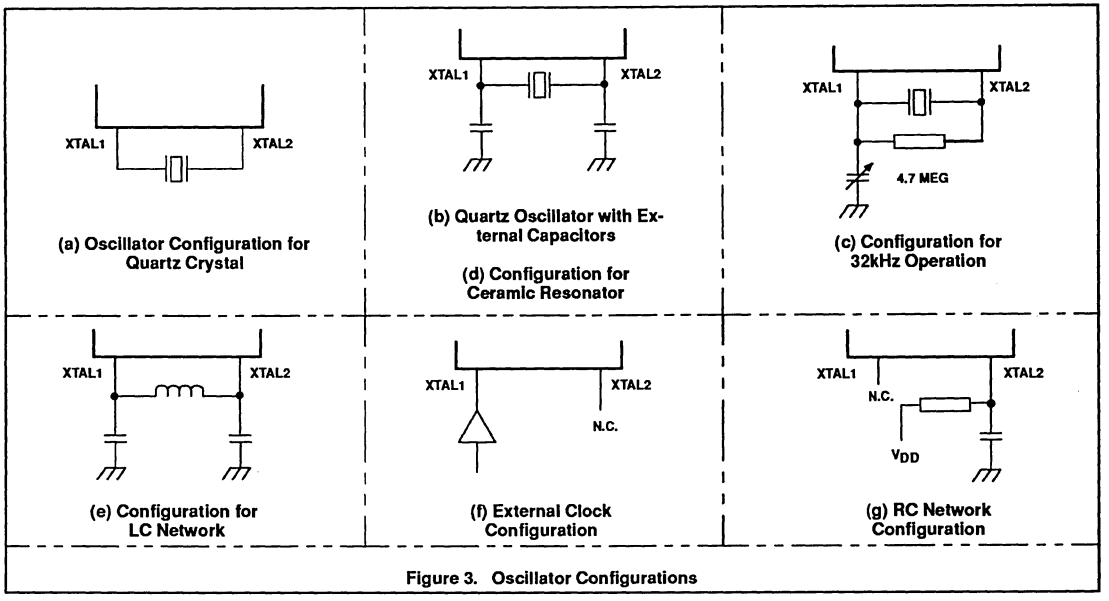


Figure 3. Oscillator Configurations

Table 2. Oscillator Type Selection Guide

RESONATOR	f (MHZ)	OPTION	C1 EXT. MIN	C1 EXT. MAX	C2 EXT. MIN	C2 EXT. MAX	MAXIMUM RESONATOR SERIES RESISTANCE
Quartz	0.032	32kHz	5	15	0	0	15kΩ[1]
Quartz	1.0	Osc.2	0	30	0	30	600Ω
Quartz	3.58	Osc.2	0	15	0	15	100Ω
Quartz	4.0	Osc.2	0	20	0	20	75Ω
Quartz	6.0	Osc.3	0	10	0	10	60Ω
Quartz	10.0	Osc.4	0	15	0	15	60Ω
Quartz	12.0	Osc.4	0	10	0	10	40Ω
Quartz	16.0	Osc.4	0	15	0	15	20Ω
PXE	0.455	Osc.2	40	50	40	50	10Ω
PXE	1.0	Osc.2	15	50	15	50	100Ω
PXE	3.58	Osc.2	0	40	0	40	10Ω
PXE	4.0	Osc.2	0	40	0	40	10Ω
PXE	6.0	Osc.2	0	20	0	20	5Ω
PXE	10.0	Osc.3	0	15	0	15	6Ω
PXE	12.0	Osc.4	10	40	10	40	6Ω
LC		Osc.2	20	90	20	90	10μH = 1Ω 100μH = 5Ω 1mH = 75Ω

NOTE:
1. 32kHz quartz crystals with a series resistance higher than 15kΩ will reduce the guaranteed supply voltage range to 2.5 to 3.5V.

Table 3. Oscillator Equivalent Circuit Parameters (see Figure 4)

PARAMETER	OPTION	SYMBOL	CONDITION	MIN	TYP	MAX	UNIT
Transconductance	32kHz	g_m	T = +25°C; V_{DD} = 4.5V	–	15	–	µs
	Osc.2	g_m	T = +25°C; V_{DD} = 4.5V	–	600	–	µs
	Osc.3	g_m	T = +25°C; V_{DD} = 4.5V	–	1500	–	µs
	Osc.4	g_m	T = +25°C; V_{DD} = 4.5V	–	4000	–	µs
Input capacitance	32kHz	$c1_i$		–	3.0	–	pF
	Osc.2	$c1_i$		–	8.0	–	pF
	Osc.3	$c1_i$		–	8.0	–	pF
	Osc.4	$c1_i$		–	8.0	–	pF
Output capacitance	32kHz	$c2_i$		–	23.0	–	pF
	Osc.2	$c2_i$		–	8.0	–	pF
	Osc.3	$c2_i$		–	8.0	–	pF
	Osc.4	$c2_i$		–	8.0	–	pF
Output resistance	32kHz	R2		–	3800	–	kΩ
	Osc.2	R2		–	65	–	kΩ
	Osc.3	R2		–	18	–	kΩ
	Osc.4	R2		–	5.0	–	kΩ

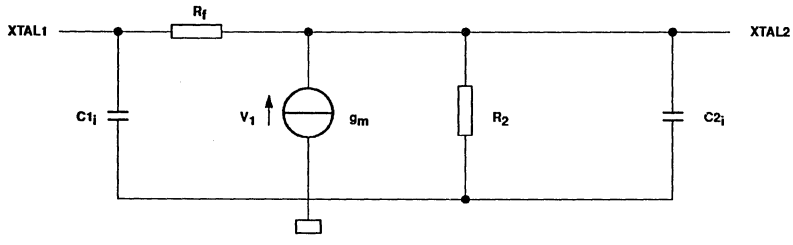


Figure 4. Equivalent Circuit Diagram

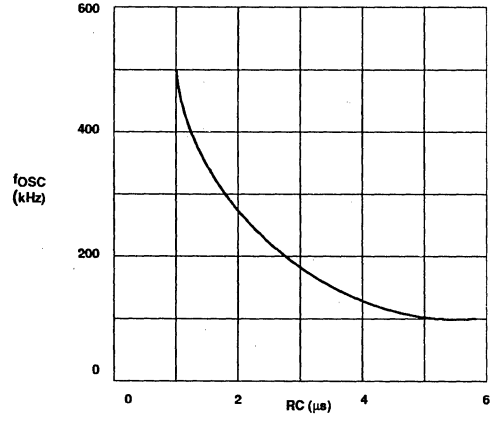


Figure 5. Frequency as a Function of RC

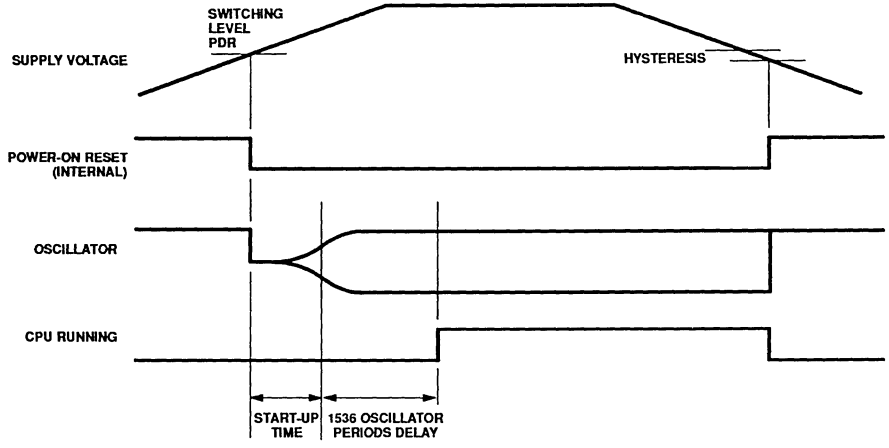


Figure 6. Power-on Reset Switching Level

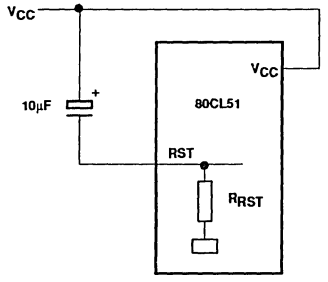


Figure 7. Recommended Power-on Reset Circuitry

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Supply voltage	−0.5 to +6.5	V
All input voltages	−0.5 to V_{DD} +0.5	V
DC current into any input or output	5	mA
Total power dissipation	300	mW
Storage temperature range	−65 to +150	°C
Operating ambient temperature range	−40 to +85	°C
Operating junction temperature	125	°C

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

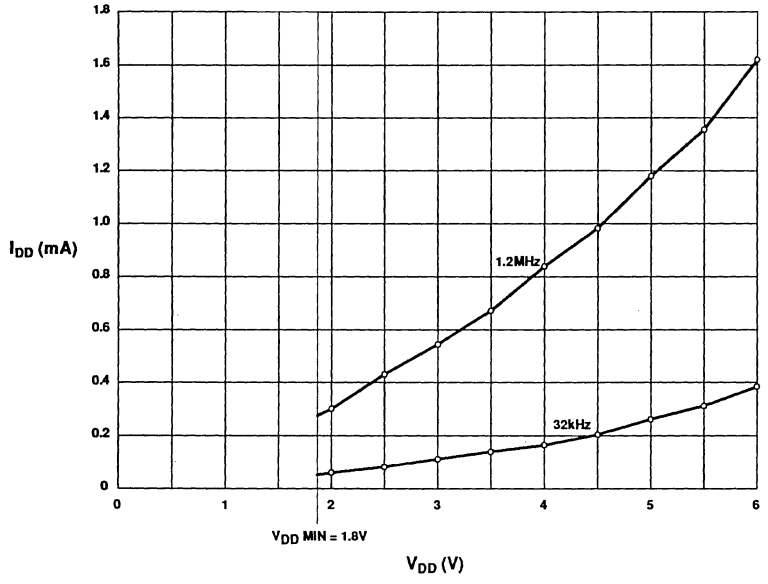
DC ELECTRICAL CHARACTERISTICS

T_{amb} = –40°C to +85°C, V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{DD}	Supply voltage	f_{CLK} (see Figure 11)	1.8	6.0	V
	RAM retention voltage in power-down mode		1.0	—	V
I_{DD}	Power supply current:				
	Operating[1]				
	OSC 1 option	f_{CLK} = 32kHz, V_{DD} = 1.8V, T_{amb} = +25°C	—	50	µA
	OSC 2 option	f_{CLK} = 3.58MHz, V_{DD} = 3V	—	2.5	mA
	OSC 2 option	f_{CLK} = 10MHz, V_{DD} = 5V	—	14	mA
	OSC 3 option	f_{CLK} = 16MHz, V_{DD} = 5V	—	18	mA
	OSC 4 option	f_{CLK} = 16MHz, V_{DD} = 5V	—	22	mA
	Idle mode[2]				
	OSC 1 option	f_{CLK} = 32kHz, V_{DD} = 1.8V, T_{amb} = +25°C	—	25	µA
	OSC 2 option	f_{CLK} = 3.58MHz, V_{DD} = 3V	—	1.0	mA
	OSC 2 option	f_{CLK} = 10MHz, V_{DD} = 5V	—	5.0	mA
	OSC 3 option	f_{CLK} = 16MHz, V_{DD} = 5V	—	7.5	mA
	OSC 4 option	f_{CLK} = 16MHz, V_{DD} = 5V	—	9.0	mA
	Power-down mode[3]	V_{DD} = 1.8V, T_{amb} = +25°C	—	10	µA
V_{IL}	Input low voltage		V_{SS}	0.3V_{DD}	V
V_{IH}	Input high voltage		0.7V_{DD}	V_{DD}	V
I_{OL}	Output sink current, except SDA, SCL	V_{DD} = 5V, V_{OL} = 0.4V	1.6		mA
		V_{DD} = 2.5V, V_{OL} = 0.4V	0.7		mA
I_{OL1}	Output sink current, SDA, SCL	V_{DD} = 5V, V_{OL} = 0.4V	3.0		mA
I_{OH}	Output source current (push-pull options only)	V_{DD} = 5V, V_{OH} = V_{DD} – 0.4V	1.6		mA
		V_{DD} = 2.5V, V_{OH} = V_{DD} – 0.4V	0.7		mA
I_{IL}	Logical 0 input current, ports 1, 2, 3	V_{DD} = 5V, V_{IN} = 0.4V		–100	µA
		V_{DD} = 2.5V, V_{IN} = 0.4V		–50	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3	V_{DD} = 5V, V_{IN} = V_{DD}/2		–1.0	mA
		V_{DD} = 2.5V, V_{IN} = V_{DD}/2		–500	µA
I_{LI}	Input leakage current, port 0, $\overline{EA}$, SCL, SDA	V_{SS} < V_I < V_{DD}		±10	µA
R_{RST}	Internal reset pull-down resistor		10	200	kΩ

NOTES:

1. The operating supply current is measured with all output pins disconnected; XTAL1 driven with t_r = t_f = 10ns; V_{IL} = V_{SS}, V_{IH} = V_{DD}; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = V_{DD}; all open drain outputs connected to V_{SS}.
2. The idle supply current is measured with all output pins disconnected; XTAL1 driven with t_r = t_f = 10ns; V_{IL} = V_{SS}, V_{IH} = V_{DD}; XTAL2 not connected; $\overline{EA}$ = Port 0 = V_{DD}; RST = V_{SS}; all open drain outputs connected to V_{SS}.
3. The power-down current is measured with all output pins disconnected; XTAL1 not connected; $\overline{EA}$ = port 0 = V_{DD}; RST = V_{SS}; all open-drain outputs connected to V_{SS}.
4. Circuits with option "no power-on reset" are tested at V_{DDMIN} = 1.8V, with option POR = 1.3V at V_{DDMIN} = 2.5V.



Typical Operating Current Versus Supply and Frequency (32kHz–1.2MHz) at +25°C

AC ELECTRICAL CHARACTERISTICS

T_{amb} = –40°C to +85°C, V_{SS} = 0V[1, 2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
Program Memory							
$1/t_{CLCL}$		Oscillator frequency			0	20	MHz
t_{LL}	8	ALE pulse width	127		$2t_{CLCL}$–40		ns
t_{AL}	8	Address valid to ALE low	43		t_{CLCL}–40		ns
t_{LA}	8	Address hold after ALE low	48		t_{CLCL}–35		ns
t_{LIV}	8	ALE low to valid instruction in		233		$4t_{CLCL}$–100	ns
t_{LC}	8	ALE low to $\overline{PSEN}$ low	58		t_{CLCL}–25		ns
t_{CC}	8	$\overline{PSEN}$ pulse width	215		$3t_{CLCL}$–35		ns
t_{CIV}	8	$\overline{PSEN}$ low to valid instruction in		125		$3t_{CLCL}$–125	ns
t_{CI}	8	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{CIF}	8	Input instruction float after $\overline{PSEN}$		63		t_{CLCL}–20	ns
t_{AVI}	8	Address to valid instruction in		302		$5t_{CLCL}$–115	ns
t_{AFC}	8	$\overline{PSEN}$ low to address float	0		0		ns
Data Memory							
t_{RR}	9	$\overline{RD}$ pulse width	400		$6t_{CLCL}$–100		ns
t_{WW}	10	$\overline{WR}$ pulse width	400		$6t_{CLCL}$–100		ns
t_{LA}	9, 10	Address hold time after ALE	48	–	t_{CLCL}–35	–	ns
t_{RD}	9	$\overline{RD}$ low to valid data in		250		$5t_{CLCL}$–165	ns
t_{DFR}	9	Data float after $\overline{RD}$		97		$2t_{CLCL}$–70	ns
t_{LD}	9	ALE low to valid data in		517		$8t_{CLCL}$–150	ns
t_{AD}	9	Address to valid data in		585		$9t_{CLCL}$–165	ns
t_{LW}	9, 10	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}$–50	$3t_{CLCL}$+50	ns
t_{AW}	9, 10	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}$–130		ns
t_{DWX}	10	Data valid to $\overline{WR}$ transition	23		t_{CLCL}–60		ns
t_{DW}	9	Data valid to $\overline{WR}$	433	–	$7t_{CLCL}$–150	–	ns
t_{WD}	10	Data hold after $\overline{WR}$	33		t_{CLCL}–50		ns
t_{AFR}	9	$\overline{RD}$ low to address float[3]		12		12	ns
t_{WHLH}	9, 10	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	t_{CLCL}–40	t_{CLCL}+40	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 50pF, load capacitance for all other outputs = 40pF.
3. Interfacing the 80CL51 to devices with float time up to 75ns is permitted. This limited bus connection will not cause damage to port 0 drivers.

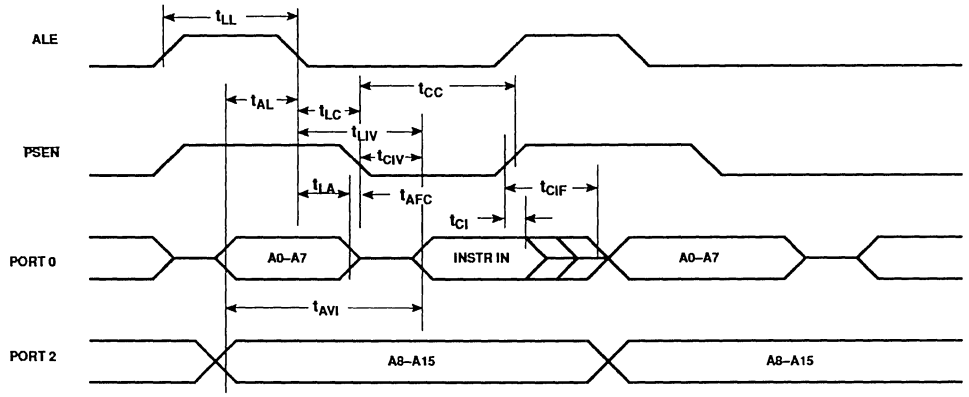


Figure 8. External Program Memory Read Cycle

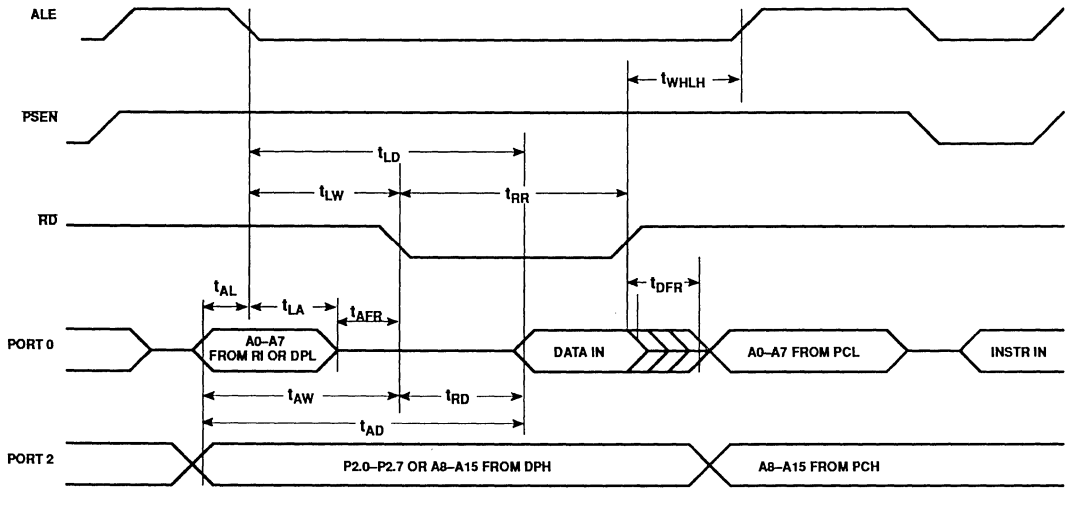


Figure 9. External Data Memory Read Cycle

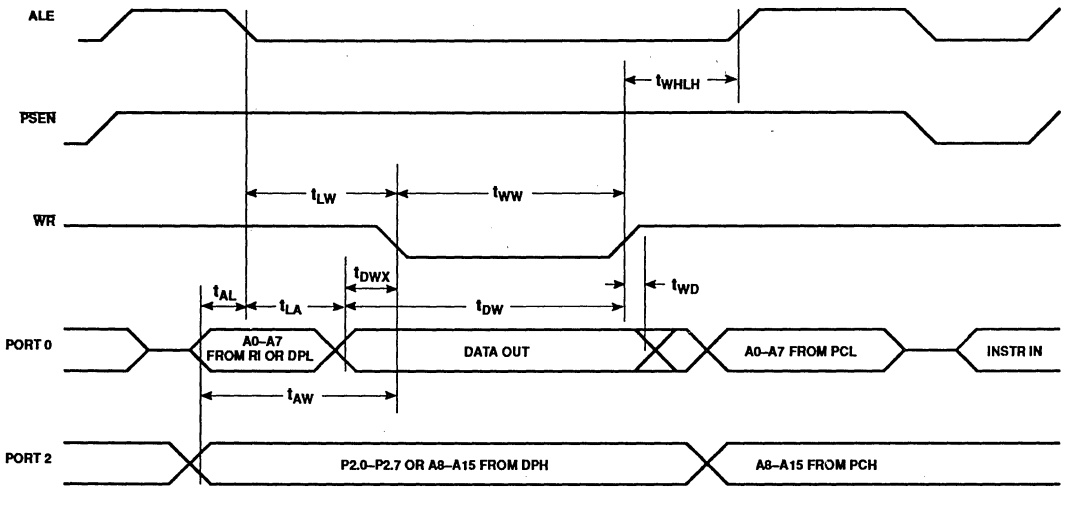


Figure 10. External Data Memory Write Cycle

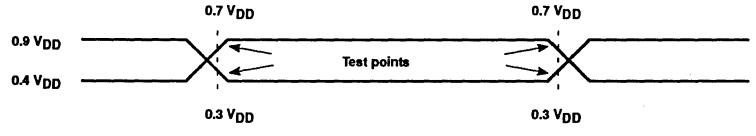


Figure 11. AC Testing Input Waveform

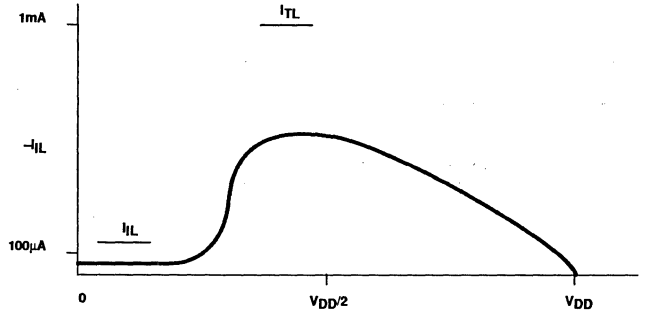
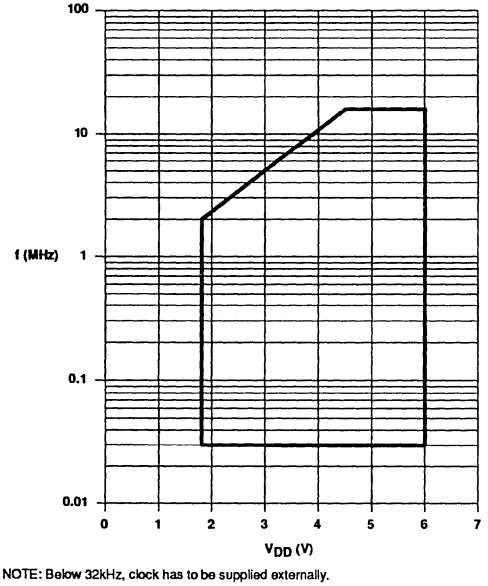


Figure 12. Input Current



NOTE: Below 32kHz, clock has to be supplied externally.

Figure 13. Frequency Operating Range

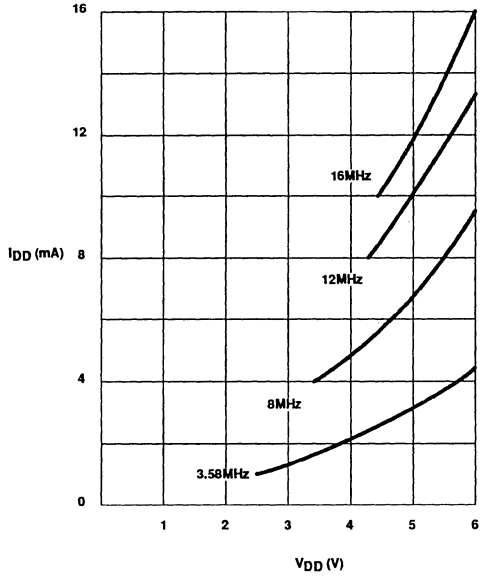


Figure 14. Typical Operating Current as a Function of Frequency and V_{DD}, T_{amb} = 25°C

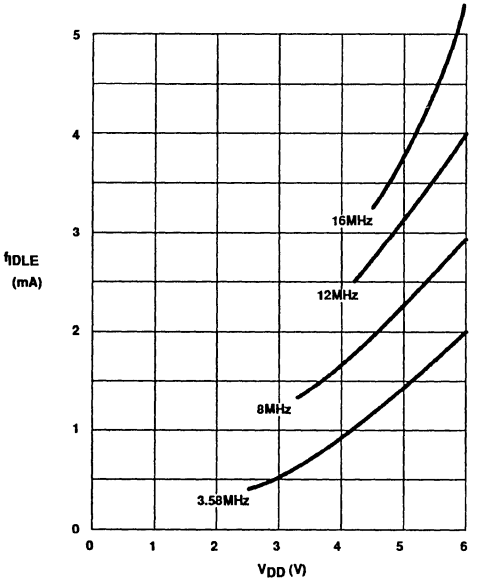


Figure 15. Typical Idle Current as a Function of Frequency and V_{DD}, T_{amb} = 25°C

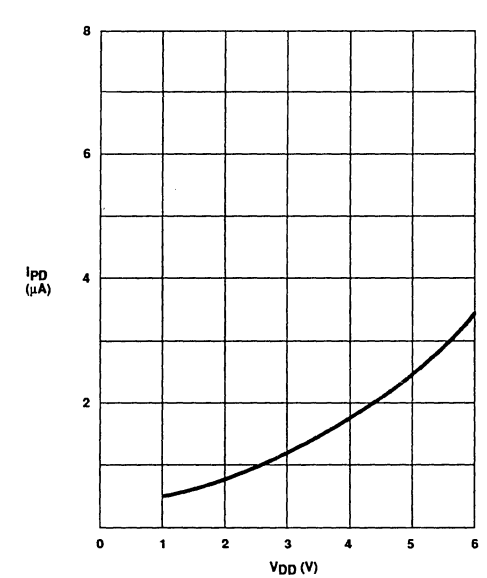


Figure 16. Typical Power-Down Current Vs. Frequency and V_{DD}, T_{amb} = 25°C

Low voltage/low power single-chip 8-bit microcontroller with UART

80CL31/80CL51

PIGGYBACK SPECIFICATION

The differences between the masked version and the piggyback are described herein.

General Description

The P85CL000HFZ is a piggy-back version with 256 bytes of RAM used for emulation of the P80CL51 microcontroller. The P85CL000HFZ is manufactured in an advanced CMOS technology. The instruction set of the P85CL000HFZ is based on that of the 8051. The device has low power consumption and a wide supply voltage range. The P85CL000HFZ has two software selectable modes of reduced activity for further power reduction: Idle and Power-down. For timing and AC/DC characteristics, please refer to the P80CL51 specifications.

Features

- Full static 80C51 CPU
- 8-bit CPU, RAM, I/O in a single 40-lead DIP
- Socket for up to 16k external EPROM
- 256 bytes RAM, expandable externally to 64K bytes
- Four 8-bit ports, 32 I/O lines
- Two 16-bit timer/event counters
- External memory expandable up to 128K, external ROM up to 64K and/or RAM up to 64K
- Thirteen source, thirteen vector interrupt structure with two priority levels
- Full duplex serial port (UART)
- I^2C-bus interface for serial transfer on two lines
- Enhanced architecture with:
  - non-page oriented instructions
  - direct addressing
  - four eight byte RAM register banks
  - stack depth up to 128 bytes
  - multiply, divide, subtract and compare instructions
- STOP and IDLE instructions
- Wake-up via external interrupts at port 1
- Single supply voltage of 1.8V to 6.0V
- On-chip oscillator (option: oscillator 4)
- Very low current consumption
- Operating temperature range: –40 to +85°C

STANDARD PIGGYBACK

Types: P85CL000HFZ
Emulation for: P83CL410, P80CL51
List of differences between masked microcontroller and corresponding piggyback:

PARAMETER	MASKED CONTROLLER	PIGGYBACK
RAM size	128	256
ROM size	4k	EPROM size dependent (max 16k)
Port option	1, 2, 3	1
Oscillator option	32kHz, Osc, 2, 3, 4, RC	Osc. 4
Mech. dimensions	Standard Dual In-Line, Small Outline	See Figure 17
Current cons.	I_{DD}	I_{DD} (OSC. 4) + I_{EPROM}
Voltage range	full	full, limited by EPROM
ESD	specification	not tested (different package)

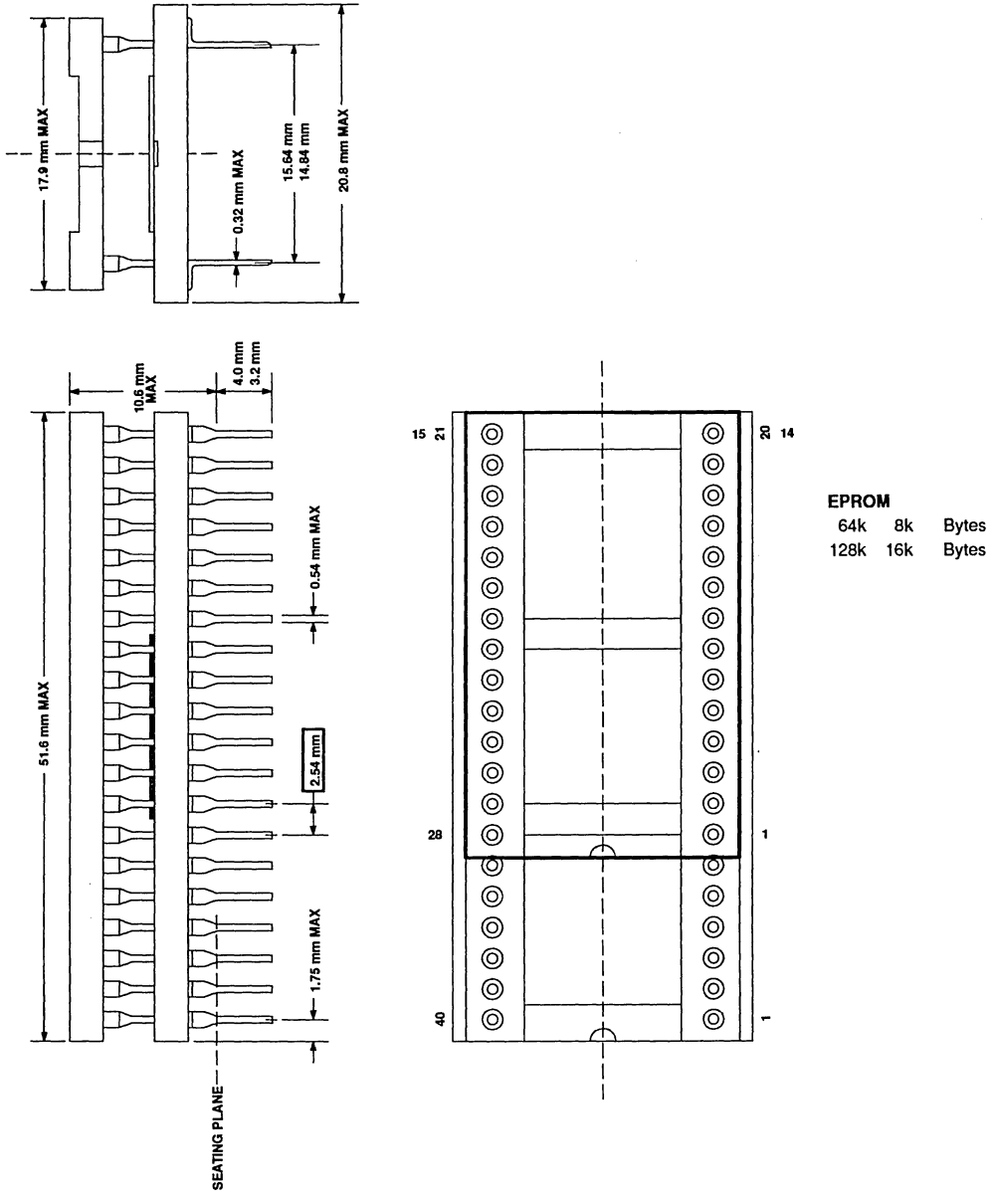


Figure 17. Piggyback Dimensions
40-Lead Dual In-Line Piggyback: P85CL000HFZ

8XC52 OVERVIEW

The 8XC52 is identical to the 8XC51, respectively, except for added features. The 8XC52 has:

- 8k ROM (80C52 only)
- 256 bytes RAM
- Counter/timer 2

As a result, there are some additions to the interrupt structure, I/O pin alternate functions, and baud rate generation for the serial channel.

Since the similarity is so close, only the additional features of the 8XC52 are described. Where necessary, some repetition of 80C51 information will be made for the sake of clarity. The 8XC52 is pin for pin and fully code compatible with the 80C51.

Differences From the 80C51

Program Memory

The data and program memory are organized virtually identically to the 80C51. The 80C52 possesses 8k bytes of on-chip program memory. When $\overline{EA}$ is high, the 80C52 fetches instructions from the internal ROM unless the address exceeds 1FFFh. Locations 2000H to FFFFH are fetched from external program memory. When $\overline{EA}$ is held low all instruction fetches are from external memory. The program memory space is shown in Figure 1.

The data memory organization is identical to the 80C51 except that the 80C52 has an additional 128 bytes of RAM overlapped with the special functio register space. This additional RAM is addressed using indirect addressing only and is available as stack space. The 80C52 data memory space is shown in Figure 2.

Special Function Registers

The special function register space is the same as the 80C51 except that the 80C52 contains the additional special function registers T2CON, RCAP2L, RCAP2H, TL2, and TH2. Since the standard 80C51 on-chip functions are identical in the 80C52, the SFR locations, bit locations, and operation are likewise identical. The only exceptions are in the interrupt mode and interrupt priority SFRs (see Table 1).

Timer/Counters

In addition to timer/counters 0 and 1 of the 80C51, the 80C52 contains timer/counter 2. Like timers 0 and 1, timer 2 can operate as either an event timer or as an event counter. This is selected by bit C/T2 in the special function register T2CON (see Figure 3). It has three operating modes: capture, auto-load, and baud rate generator, which are selected by bits in the T2CON as shown in Table 2.

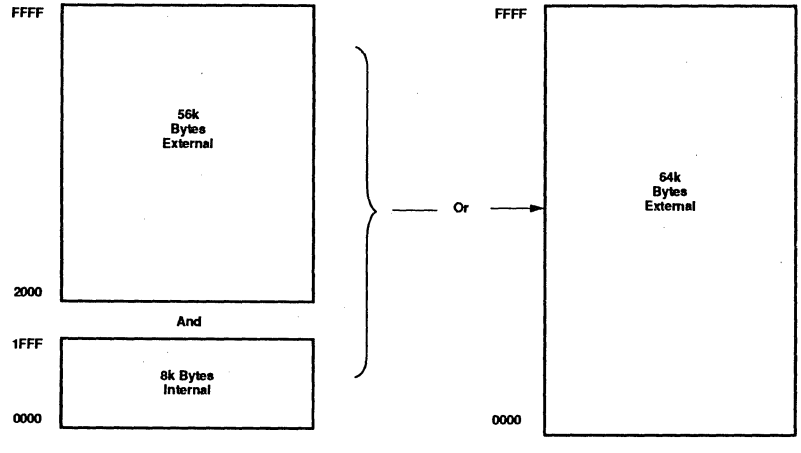


Figure 1. 80C52 Program Memory

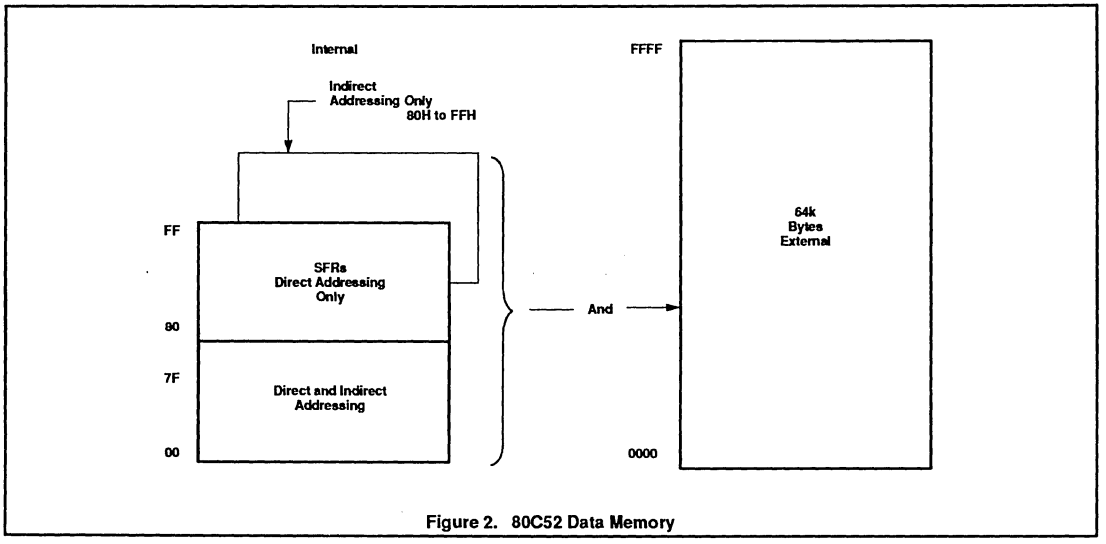


Figure 2. 80C52 Data Memory

In the Capture Mode there are two options which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then Timer 2 is a 16-bit timer or counter which upon overflowing sets bit TF2, the Timer 2 overflow bit, which can be used to generate an interrupt. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX causes the current value in the Timer 2 registers, TL2 and TH2, to be captured into registers RCAP2L and RCAP2H, respectively. (RCAP2L and RCAP2H are new special function registers in the 80C52.) In addition, the transition at T2EX causes bit EXF2 in T2CON to be set, and EXF2 like TF2 can generate an interrupt. The Capture Mode is illustrated in Figure 4.

In the auto-reload mode, there are again two options, which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then when Timer 2 rolls over it not only sets TF2 but also causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2L and RCAP2H, which are preset by software. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX will also trigger the 16-bit reload and set EXF2. The auto-reload mode is illustrated in Figure 5.

The baud rate generation mode is selected by RCLK = 1 and/or TCLK = 1. It will be described in conjunction with the serial port.

Serial Port

The serial port of the 8XC52 is identical to that of the 80C51 except that counter/timer 2 can be used to generate baud rates.

In the 80C52, Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (see Figure 3). Note that the baud rate for transmit and receive can be simultaneously different. Setting RCLK and/or TCLK puts Timer into its baud rate generator mode, as shown in Figure 6.

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

Now, the baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate as follows:

$$\text{Modes 1, 3 Baud Rate} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The timer can be configured for either "timer" or "counter" operation. In the most typical applications, it is configured for "timer" operation (C/T2 = 0). "Timer" operation is a little different for Timer 2 when it's being used as a baud rate generator. Normally, as a timer it would increment every machine cycle (thus at 1/12 the oscillator frequency). As a baud rate generator, however, it increments every state time (thus at 1/2 the oscillator frequency). In that case the baud rate is given by the formula:

$$\text{Modes 1, 3 Baud Rate} = \frac{\text{Oscillator Frequency}}{32\times[65536 - (\text{RCAP2H, RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

(MSB) (LSB)

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2

Symbol	Position	Name and Significance
TF2	T2CON.7	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK or TCLK = 1.
EXF2	T2CON.6	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software.
RCLK	T2CON.5	Receive clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	T2CON.4	Transmit clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	T2CON.3	Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	T2CON.2	Start/stop control for Timer 2. A logic 1 starts the timer.
C/T2	T2CON.1	Timer or counter select. (Timer 2) 0 = Internal timer (OSC/12) 1 = External event counter (falling edge triggered).
CP/RL2	T2CON.0	Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

Figure 3. Timer/Counter 2 (T2CON) Control Register

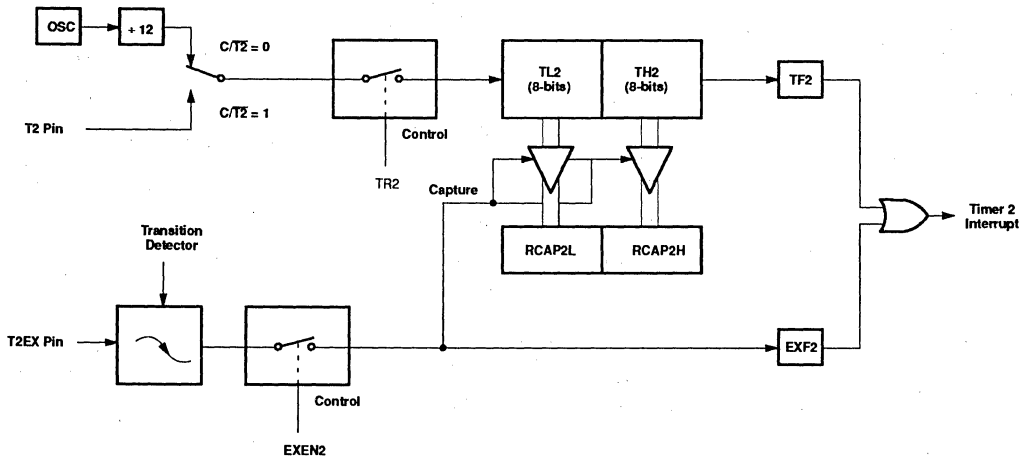


Figure 4. Timer 2 in Capture Mode

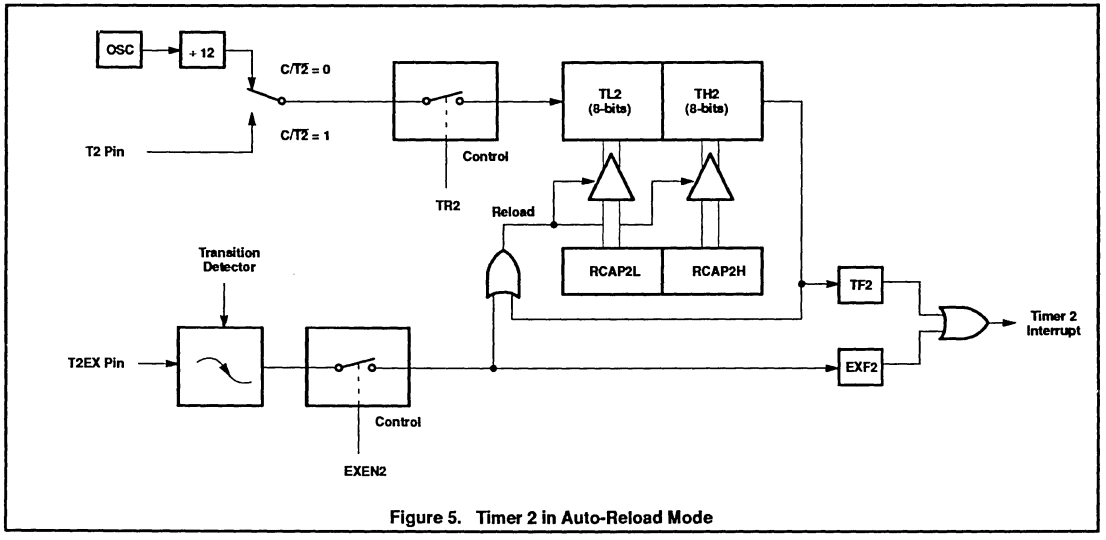


Figure 5. Timer 2 in Auto-Reload Mode

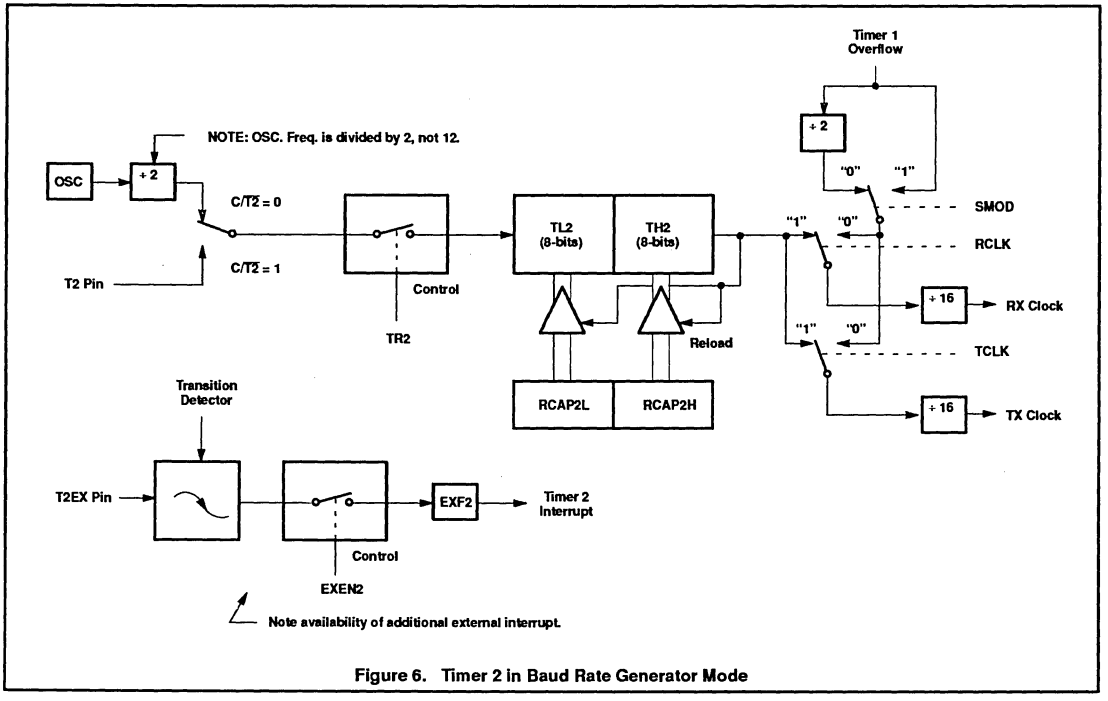


Figure 6. Timer 2 in Baud Rate Generator Mode

Table 1. 8XC52 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt enable	A8H	EA	–	ET2	ES	ET1	EX1	ET0	EX0	0x000000B
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt priority	B8H	–	–	PT2	PS	PT1	PX1	PT0	PX0	xx000000B
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	–	–	–	–	–	–	T2EX	T2	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	A15	A14	A13	A12	A11	A10	A9	A8	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	$\overline{RD}$	$\overline{WR}$	T1	T0	$\overline{INT1}$	$\overline{INT0}$	TxD	RxD	FFH
PCON[1]	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxxxxxxB
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
RCAP2H#	Capture high	CBH									00H
RCAPL#	Capture low	CAH									00H
SBUF	Serial data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial controller	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
			CF	CE	CD	CC	CB	CA	C9	C8	
T2CON*#	Timer 2 control	C8H	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$	00H
TH0	Timer high 0	8CH									00H
TH1	Timer high 1	8DH									00H
TH2#	Timer high 2	CDH									00H
TL0	Timer low 0	8AH									00H
TL1	Timer low 1	8BH									00H
TL2#	Timer low 2	CCH									00H
TMOD	Timer mode	89H	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	00H

* Bit addressable
\# SFRs are modified from or added to the 80C51 SFRs.
1 Bits GF1, GF0, PD, and IDL of the PCON register are not implemented in the NMOS 8XC52.

Table 2. Timer 2 Operating Modes

RCLK + RCLK	CP/RL2	TR2	MODE
0	0	1	16-bit Auto-reload
0	1	1	16-bit Capture
1	X	1	Baud rate generator
X	X	0	(off)

Timer 2 as a baud rate generator is shown in Figure 6. This figure is valid only if RCLK + TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Therefore, the Timer 2 interrupt does not have to be disabled when Timer 2 is in the baud rate generator mode. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt, if desired.

It should be noted that when Timer 2 is running (TR2 = 1) in "timer" function in the baud rate generator mode, one should not try to read or write TH2 or TL2. Under these conditions the timer is being incremented every state time, and the results of a read or write may not be accurate. The RCAP registers may be read, but should not be written to, because a write might overlap a reload and cause write and/or reload errors. Turn the timer off (clear TR2) before accessing the Timer 2 or RCAP registers, in this case.

The serial port in Modes 1 and 3 with the timer 2 baud rate interface is shown in Figures 7 and 8.

Timer/Counter 2 Set-up

Except for the baud rate generator mode, the values given for T2CON do not include the setting of the TR2 bit. Therefore, bit TR2 must be set, separately, to turn the timer on. See Table 3 for set-up of timer 2 as a timer. See Table 4 for set-up of timer 2 as a counter.

Using Timer/Counter 2 to Generate Baud Rates

For this purpose, Timer 2 must be used in the baud rate generating mode. If Timer 2 is being clocked through pin T2 (P1.0) the baud rate is:

$$\text{Baud Rate} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

And if it is being clocked internally, the baud rate is:

$$\text{Baud Rate} = \frac{\text{Oscillator Frequency}}{32\text{x}[65536 - (\text{RCAP2H, RCA2PL})]}$$

To obtain the reload value for RCAP2H and RCA02L, the above equation can be rewritten as:

$$\text{RCAP2H, RCAP2L} = 65536 - \frac{\text{Oscillator Frequency}}{32 \times \text{Baud Rate}}$$

Interrupts

The 80C52 has 6 interrupt sources as shown in Figure 9. All except TF2 and EXF2 are identical sources to those in the 80C51.

The Interrupt Enable Register and the Interrupt Priority Register are modified to include the additional 80C52 interrupt sources. The operation of these registers is identical to the 80C51. The registers are detailed in Figures 10, 11, and 12.

In the 80C52, the Timer 2 Interrupt is generated by the logical OR of TF2 and EXF2. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and the bit will have to be cleared in software.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it has been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be canceled in software.

The interrupt vector addresses and the interrupt priority for requests in the same priority level are given in the following:

Source	Vector Address	Priority Within Level
1. IE0	0003H	(highest)
2. TF0	000BH	
3. IE1	0013H	
4. TF1	001BH	
5. RI + TI	0023H	
6. TF2 + EXF2	002BH	(lowest)

Note that they are identical to those in the 80C51 except for the addition of the Timer 2 (TF1 and EXF2) interrupt at 002BH and at the lowest priority within a level.

Port Structures

The port structures are identical in both parts, except that on the 8XC52, ports P1.0 and P1.1 include the Timer 2 alternate functions as follows:

P1.0 T2 (Timer/counter 2 external input)

P1.1 T2EX (Timer/counter 2 capture/ reload trigger)

As with the 80C51, these alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1.

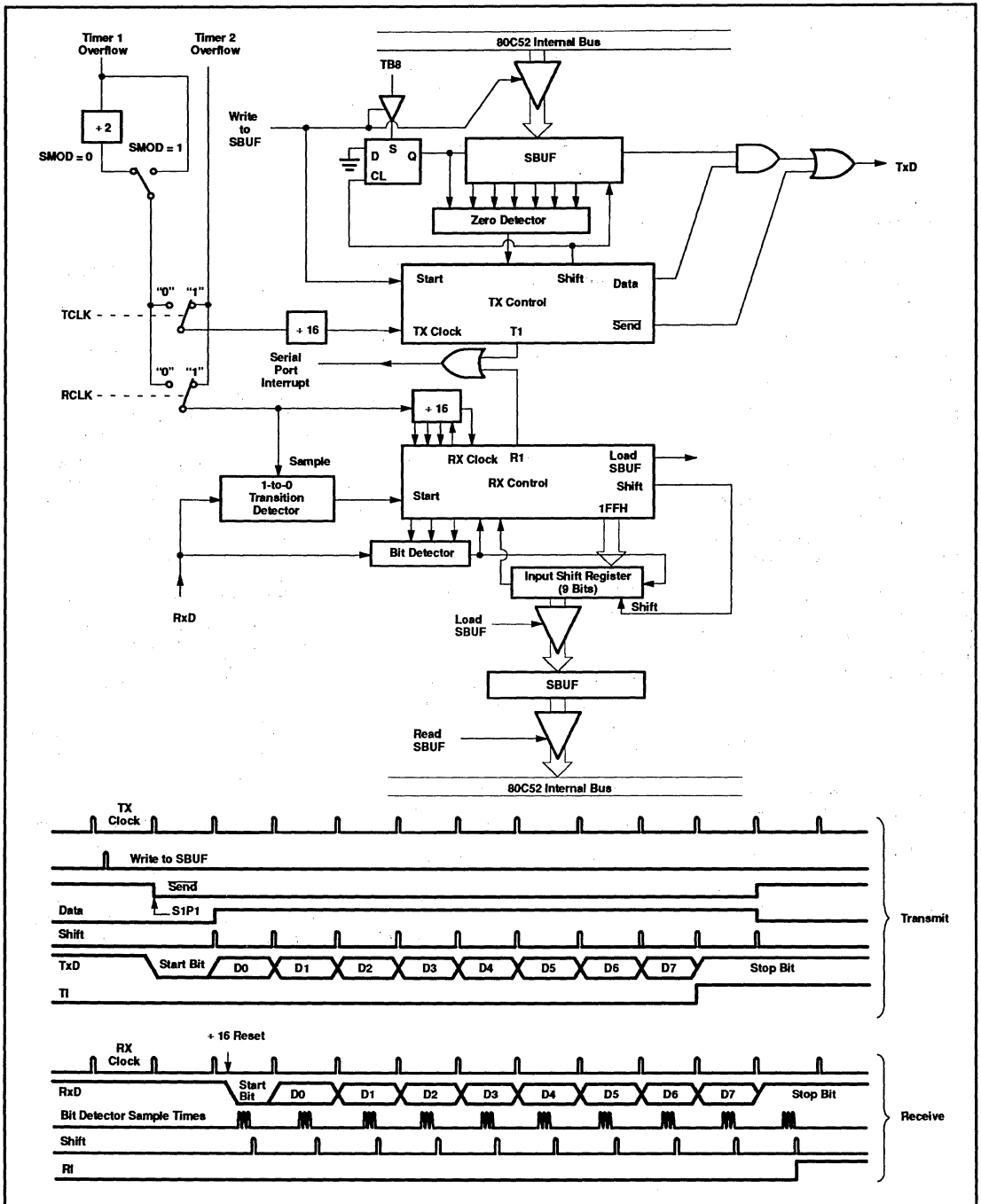


Figure 7. Serial Port Mode 1 in the 80C52

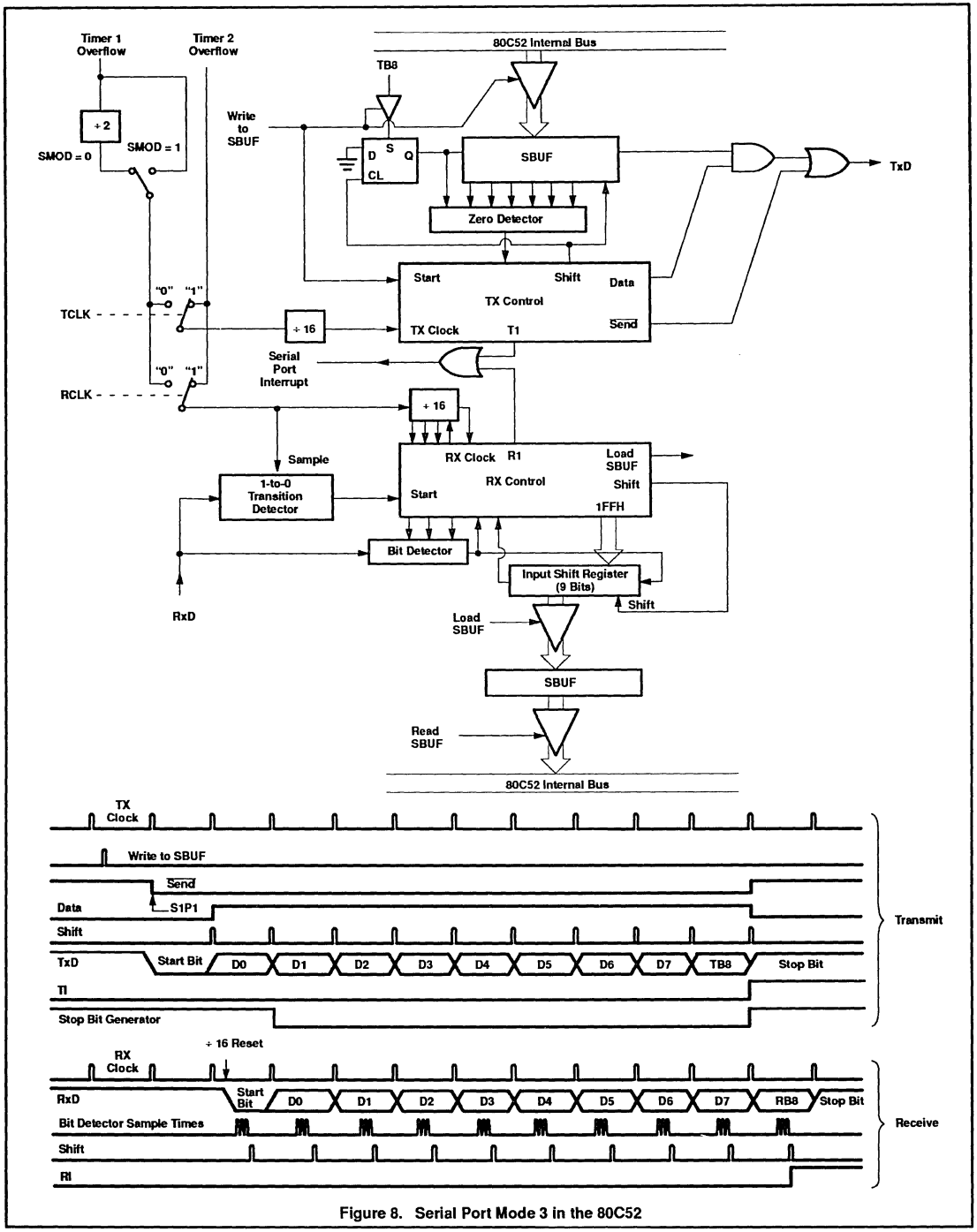


Figure 8. Serial Port Mode 3 in the 80C52

Table 3. Timer 2 as a Timer

MODE	T2CON	
	INTERNAL CONTROL (Note 1)	EXTERNAL CONTROL (Note 2)
16-bit Auto-Reload	00H	08H
16-bit Capture	01H	09H
Baud rate generator receive and transmit same baud rate	34H	36H
Receive only	24H	26H
Transmit only	14H	16H

Table 4. Timer 2 as a Counter

MODE	TMOD	
	INTERNAL CONTROL (Note 1)	EXTERNAL CONTROL (Note 2)
16-bit	02H	0AH
Auto-Reload	03H	0BH

NOTES:
1. Capture/reload occurs only on timer/counter overflow.
2. Capture/reload occurs on timer/counter overflow and a 1-to-0 transition on T2EX (P1.1) pin except when timer 2 is used in the baud rate generator mode.

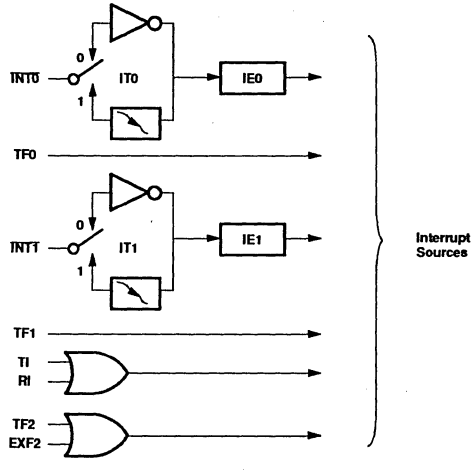


Figure 9. 80C52 Interrupt Sources

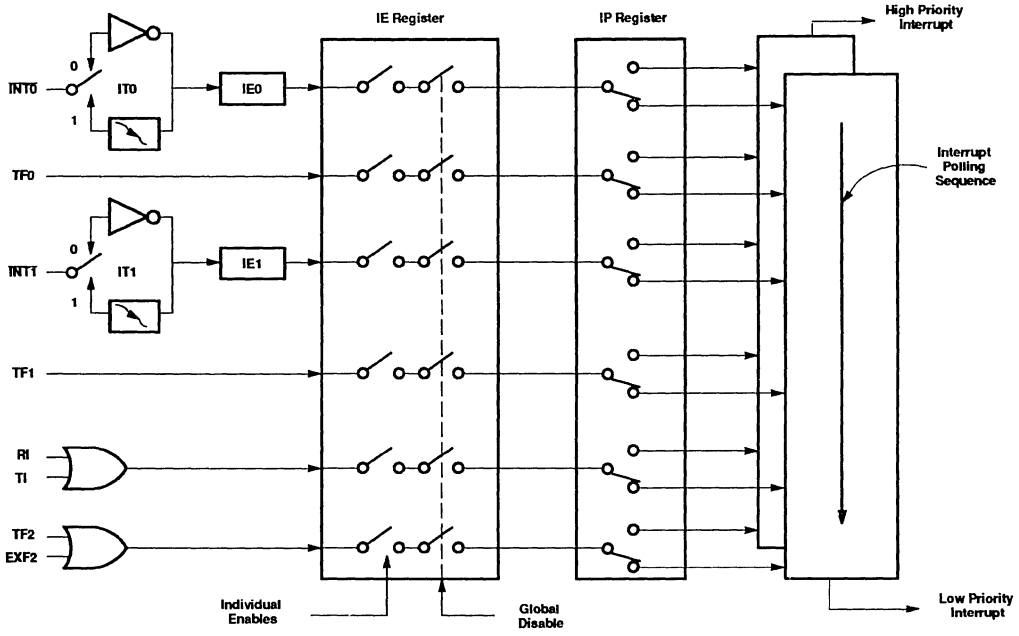


Figure 10. 80C52 Interrupt Control System

(MSB) (LSB)

EA	X	ET2	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
	IE.6	Reserved.
ET2	IE.5	Enables or disables the Timer 2 Overflow or capture interrupt. If ET2 = 0, the Timer 2 interrupt is disabled.
ES	IE.4	Enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.
ET1	IE.3	Enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	Enables or disables External Interrupt 1. If EX1 = 0, External Interrupt 1 is disabled.
ET0	IE.1	Enables or disables the Timer 0 Overflow interrupt.. If ET0 = 0, the Timer 0 interrupt is disabled.
EX0	IE.0	Enables or disables External Interrupt 0. If EX0 = 0, External Interrupt 0 is disabled.

Figure 11. 80C52 Interrupt Enable (IE) Register

(MSB) (LSB)

X	X	PT2	PS	PT1	PX1	PT0	PX0

Symbol	Position	Function
–	IP.7	Reserved.
–	IP.6	Reserved.
PT2	IP.5	Defines the Timer 2 interrupt priority level. PT2 = 1 programs it to the higher priority level.
PS	IP.4	Defines the Serial Port interrupt priority level. PS = 1 programs it to the higher priority level.
PT1	IP.3	Defines theTimer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
PX1	IP.2	Defines the External Interrupt 1 priority level. PX1 = 1 programs it to the higher priority level.
PT0	IP.1	Enables or disables the Timer 0 interrupt priority level. PT0 = 1 programs it to the higher priority level.
PX0	IP.0	Defines the External Interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

Figure 12. 80C52 Interrupt Priority (IP) Register

Single-chip 8-bit microcontroller — 8032AH/8052AH

DESCRIPTION

The Philips 8032AH/8052AH is a high-performance microcontroller fabricated using the Philips high-density highly reliable +5V, depletion-load, N-channel, silicon-gate, N500 MOS process technology. It provides the hardware features, architectural enhancements and instructions that are necessary to make it a powerful and cost-effective controller for applications requiring up to 64k bytes of program memory and/or up to 64k bytes of data storage.

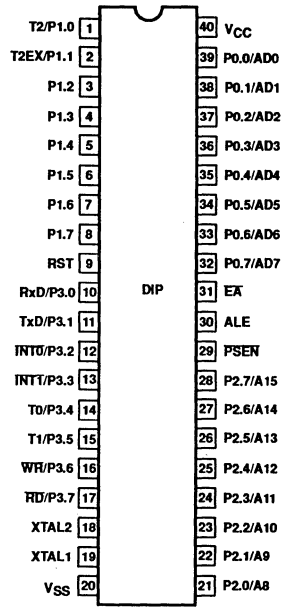
The 8032AH/8052AH contains 256 bytes of read/write data memory, 32 I/O lines configured as four 8-bit ports, three 16-bit counter/timers, a six-source, two-priority-level nested interrupt structure, a programmable serial I/O port and on-chip oscillator and clock circuitry. The 8052AH has all of these features plus 8k bytes of non-volatile read-only program memory. Both microcontrollers have memory expansion capabilities of up to 64k bytes of data storage and 64k bytes of program memory that can be attained with standard TTL compatible memories.

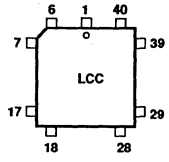
Because of its extensive BCD/binary arithmetic and bit-handling facilities, the 8032AH/8052AH microcontroller is efficient at both computational and control-oriented tasks. Efficient use of program memory is also achieved by using the familiar compact instruction set of the 8031AH/8051AH. Forty-four percent of the instructions are one-byte, 41% two-byte, and 15% three-byte instructions. With a 12MHz crystal, the majority of the instructions execute in just 1.0µs. The longest instructions, multiply and divide, require only 4µs at 12MHz.

FEATURES

- 8032AH — control-oriented CPU with RAM and I/O
- 8052AH — an 8032AH with factory mask-programmable ROM
- 8k X 8 ROM (8052AH only)
- 256 X 8 RAM
- Four 8-bit ports, 32 I/O lines
- Three 16-bit timer/counters
- Programmable full-duplex serial channel
  - Variable transmit/receive baud rate capability
- Timer 2 capture capability
- External memory
  - 64k ROM and 64k RAM
- Boolean processor
- 128 user bit-addressable locations
- Upward compatible with 8031AH/8051AH

PIN CONFIGURATIONS



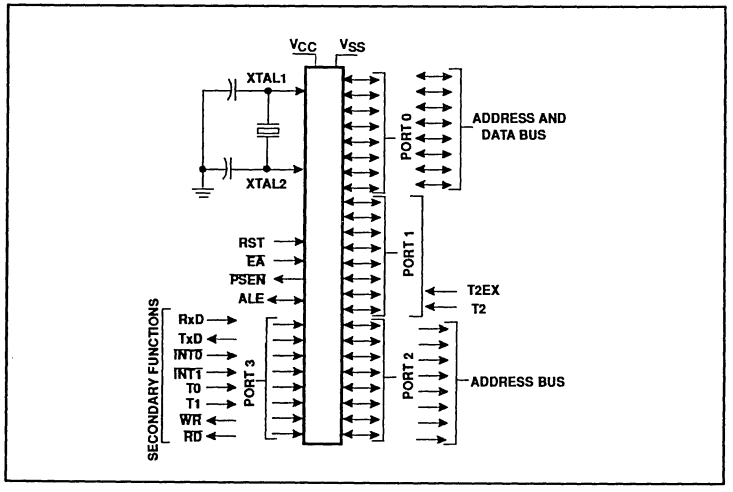


Pin	Function	Pin	Function	Pin	Function
1	NC	15	INT1/P3.3	30	P2.6/A14
2	T2/P1.0	16	T0/P3.4	31	P2.7/A15
3	T2EX/P1.1	17	T1/P3.5	32	PSEN
4	P1.2	18	WR/P3.6	33	ALE
5	P1.3	19	RD/P3.7	34	NC
6	P1.4	20	XTAL2	35	EA
7	P1.5	21	XTAL1	36	P0.7/AD7
8	P1.6	22	V_{SS}	37	P0.6/AD6
9	P1.7	23	NC	38	P0.5/AD5
10	RST	24	P2.0/A8	39	P0.4/AD4
11	RxD/P3.0	25	P2.1/A9	40	P0.3/AD3
12	NC	26	P2.2/A10	4'1	P0.2/AD2
13	TxD/P3.1	27	P2.3/A11	42	P0.1/AD1
14	INT0/P3.2	28	P2.4/A12	43	P0.0/AD0
		29	P2.5/A13	44	V_{CC}

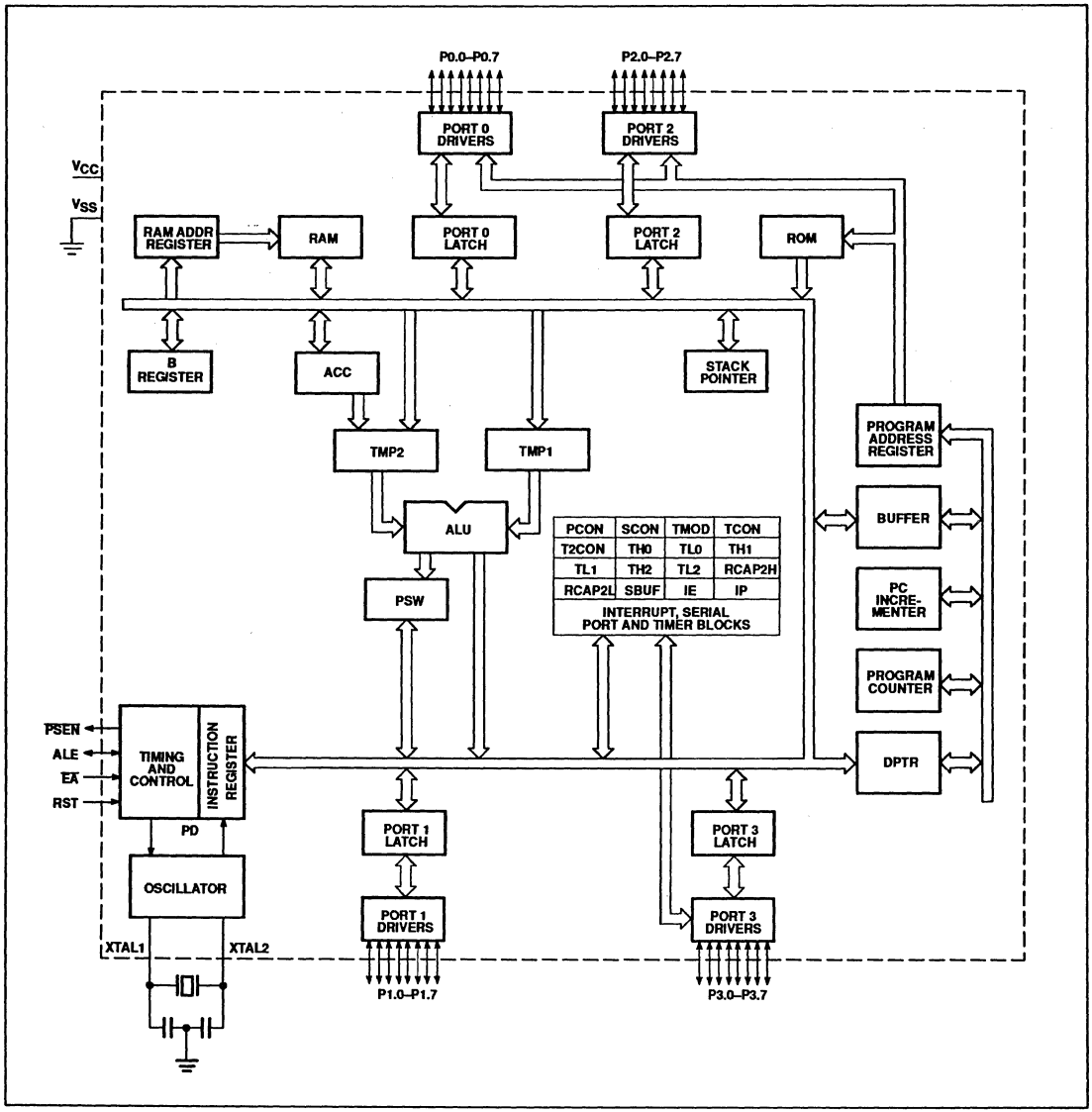
PART NUMBER SELECTION

PHILIPS		SIGNETICS		TEMPERATURE °C AND PACKAGE	FREQUENCY MHz
ROMless	ROM	ROMless	ROM		
MAB8032AH-2P	MAB8052AH-2P	SCN8032HCCN40	SCN8052HCCN40	0 to +70, plastic DIP	12
MAB8032AH-2WP	MAB8052AH-2WP	SCN8032HCCA44	SCN8052HCCA44	0 to +70, plastic LCC	12
MAF8032AH-2P	MAF8052AH-2P	SCN8032HACN40	SCN8052HACN40	–40 to +85, plastic DIP	12
MAF8032AH-2WP	MAF8052AH-2WP	SCN8032HACA44	SCN8052HACA44	–40 to +85, plastic LCC	12
		SCN8032HCFN40	SCN8052HCFN40	0 to +70, plastic DIP	15
		SCN8032HCFA44	SCN8052HCFA44	0 to +70, plastic LCC	15
		SCN8032HAFN40	SCN8052HAFN40	–40 to +85, plastic DIP	15
		SCN8032HAFA44	SCN8052HAFA44	–40 TO +85, plastic LCC	15

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	20	22	I	**Ground:** 0V reference.
V_{CC}	40	44	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–36	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s.
P1.0–P1.7	1–8	2–9	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Pins P1.0 and P1.1 also correspond to the special functions T2, timer 2 counter trigger input, and T2EX, external input to timer 2. the output latch on these two special functions must be programmed to one for that function to operate. Port 1 also receives the low-order address byte during program verification.
	1	2	I	**T2 (P1.0):** Timer/counter 2 trigger input.
	2	3	I	**T2EX (P1.1):** Timer/counter 2 external count input.
P2.0–P2.7	21–28	24–31	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 also serves the special features of the SC80C51 family, as listed below:
	10	11	I	**RxD (P3.0):** Serial input port
	11	13	O	**TxD (P3.1):** Serial output port
	12	14	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	I	**T0 (P3.4):** Timer 0 external input
	15	17	I	**T1 (P3.5):** Timer 1 external input
	16	18	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. A small external pull-down resistor (approx 8.2kohm) from RST to V_{SS} permits power-on reset when a capacitor (approx 10uF) is also connected from this pin to V_{CC}.
ALE	30	33	I/O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory.
$\overline{PSEN}$	29	32	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$	31	35	I	**External Access Enable:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H and 1FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 1FFFH.
XTAL1	19	21	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	O	**Crystal 2:** Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage on any other pin to V_{SS}	–0.5 to +7.0	V
Input, output current on any single pin	10	mA
Power dissipation	1.5	W

DC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V[4, 5]

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{IL}	Input low voltage		–0.5	0.8	V
V_{IH}	Input high voltage; except XTAL2, RST		2.0	V_{CC}+0.5	V
V_{IH1}	Input high voltage to RST for reset, XTAL2	XTAL1 to V_{SS}	2.5	V_{CC}+0.5	V
V_{OL}	Output low voltage; ports 1, 2, 3[6]	I_{OL} = 1.6mA		0.45	V
V_{OL1}	Output low voltage; port 0, ALE, $\overline{PSEN}$[6]	I_{OL} = 3.2mA		0.45	V
V_{OH}	Output high voltage; ports 1, 2, 3	I_{OH} = –80uA	2.4		V
V_{OH1}	Output high voltage; port 0 in external bus mode, ALE, $\overline{PSEN}$[3]	I_{OH} = –400uA	2.4		V
I_{IL}	Logical 0 input current; ports 1, 2, 3	V_{IN} = 0.45V		–800	uA
I_{IH1}	Input high current to RST for reset	V_{IN} = V_{CC} –1.5V		500	uA
I_{LI}	Input leakage current; port 0, $\overline{EA}$	0.45 < V_{IN} < V_{CC}		±10	uA
I_{IL2}	Logical 0 input current for XTAL2	XTAL1 = V_{SS}, V_{IN} = 0.45V		–3.2	mA
I_{CC}	Power supply current	All outputs disconnected and $\overline{EA}$ = V_{CC}		175	mA
C_{IO}	Pin capacitance	f_C = 1MHz, T_{amb} = 25°C		10	pF

T_{amb} = –40°C to +85°C – Extended temperature range, V_{CC} = 5V ±10%, V_{SS} = 0V

V_{IH}	Input high voltage; except XTAL2, RST		2.2	V_{CC}+0.5	V
V_{IH1}	Input high voltage to RST for reset, XTAL2	XTAL1 to V_{SS}	2.7		V
I_{IL2}	Logical 0 input current for XTAL2	XTAL1 = V_{SS}, V_{in} = 0.45V		–3.5	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.
5. All voltage measurements are referenced to ground. For testing, all input signals swing between 0.45V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and at output voltages of 0.8V and 2.0V as appropriate.
6. V_{OL} is derated when the device rapidly discharges external capacitance. This AC noise is most pronounced during emission of address data. When using external memory, locate the latch or buffer as close as possible to the device.

Datum	Emitting Ports	Degraded I/O Lines	V_{OL} (Peak Max)
Address	P2, P0	P1, P3	0.8V
Write Data	P0	P1, p3, ALE	0.8V

7. C_L = 100pF for port 0, ALE and $\overline{PSEN}$ outputs: C_L = 80pF for all other ports.

AC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or −40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V[1,2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$		Oscillator frequency: **Speed Versions** SCN8052 C 32MAB8052/32 −2 SCN8052 F 32MAF8052/32 −2			 3.5 3.5 3.5 3.5	 12 12 15 12	 MHz MHz MHz MHz
t_{LHLL}	1	ALE pulse width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	43		$t_{CLCL}-40$		ns
t_{LLAX}	1	Address hold after ALE low	48		$t_{CLCL}-35$		ns
t_{LLIV}	1	ALE low to valid instruction in		233		$4t_{CLCL}-100$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	58		$t_{CLCL}-25$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	215		$3t_{CLCL}-35$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		125		$3t_{CLCL}-125$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		63		$t_{CLCL}-20$	ns
t_{AVIV}	1	Address to valid instruction in		302		$5t_{CLCL}-115$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		20		20	ns
t_{PXAV}	1	$\overline{PSEN}$ to address valid	75		$t_{CLCL}-8$		ns
Data Memory							
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}-165$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		97		$2t_{CLCL}-70$	ns
t_{LLDV}	2, 3	ALE low to valid data in		517		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		585		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}-130$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	23		$t_{CLCL}-60$		ns
t_{QVWH}	2, 3	Data valid to $\overline{WR}$ high	433		$7t_{CLCL}-150$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	33		$t_{CLCL}-50$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		20		20	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	5	High time	20		20		ns
t_{CLCX}	5	Low time	20		20		ns
t_{CLCH}	5	Rise time		20		20	ns
t_{CHCL}	5	Fall time		20		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

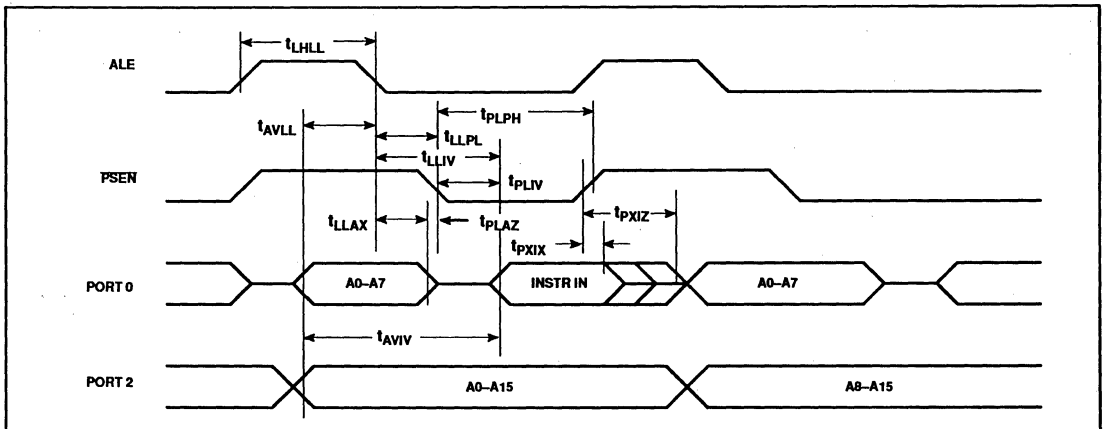


Figure 1. External Program Memory Read Cycle

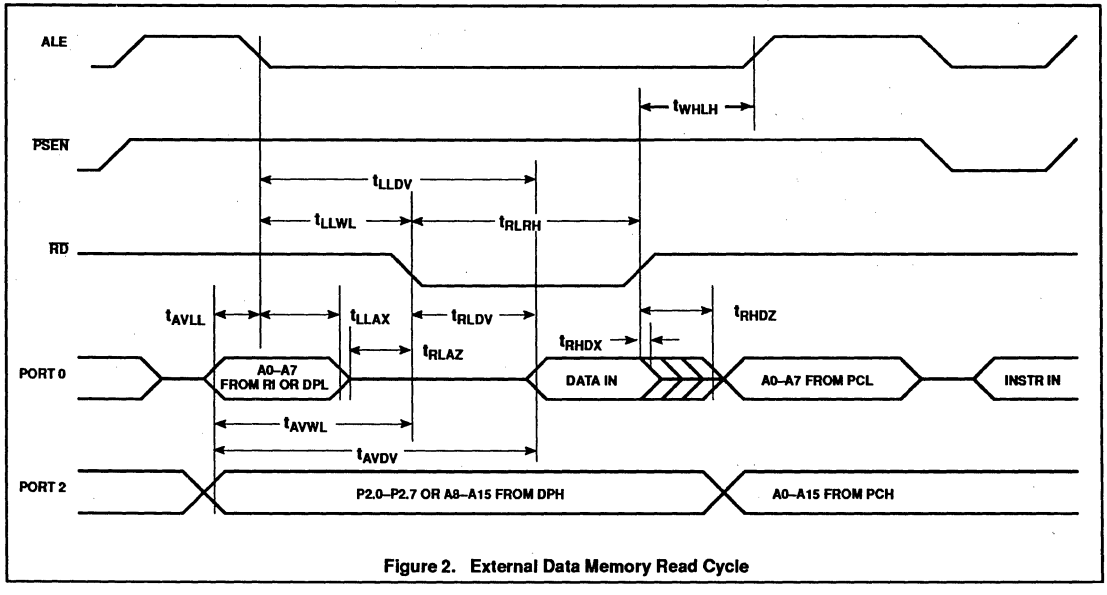


Figure 2. External Data Memory Read Cycle

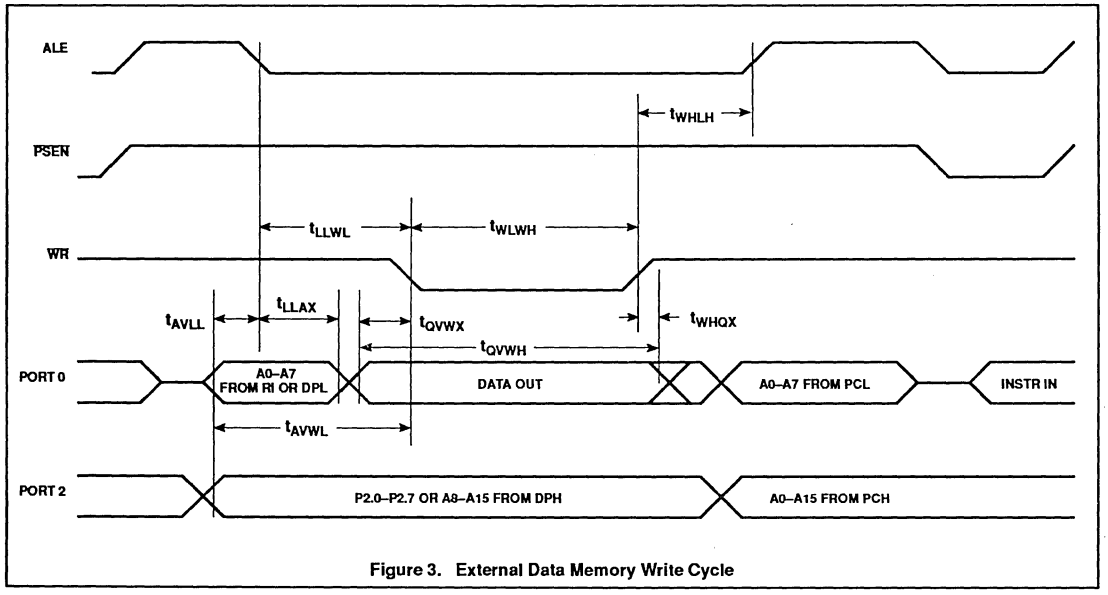


Figure 3. External Data Memory Write Cycle

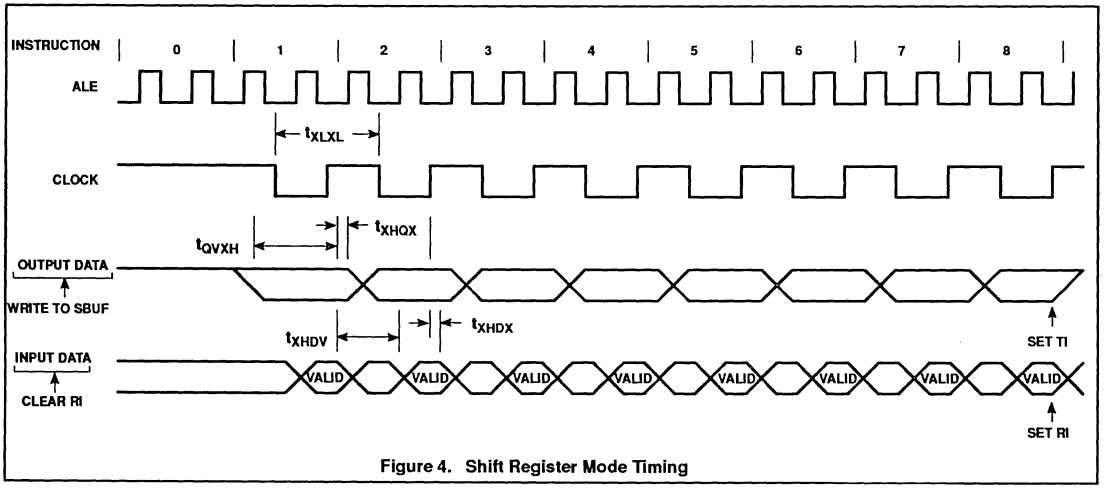


Figure 4. Shift Register Mode Timing

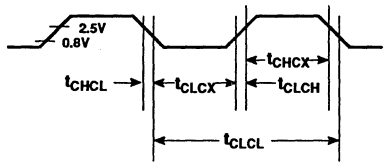
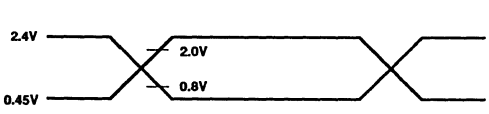
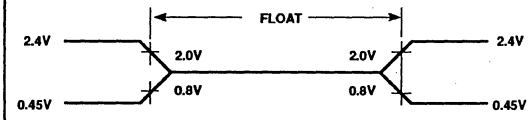


Figure 5. External Clock Drive



NOTE:
AC inputs during testing are driven at 2.4V for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at 2.0V min for a logic '1' and 0.8V for a logic '0'.

Figure 6. AC Testing Input/Output



NOTE:
For timing purposes, the float state is defined as the point at which a P0 pin sinks 2.4mA or sources 400μA at the voltage test levels.

Figure 7. Float Waveform

CMOS single-chip 8-bit microcontroller — 80C32/80C52/87C52

DESCRIPTION

The Philips 80C32/80C52/87C52 is a high-performance microcontroller fabricated with Philips high-density CMOS technology. The CMOS 8XC52 is functionally compatible with the NMOS SCN- 8032/8052 microcontrollers. The Philips CMOS technology combines the high speed and density characteristics of HMOS with the low power attributes of CMOS. Philips epitaxial substrate minimizes latch-up sensitivity.

The 8XC52 contains an 8k × 8 ROM (80C52) EPROM (87C52), a 256 × 8 RAM, 32 I/O lines, three 16-bit counter/timers, a six-source, two-priority level nested interrupt structure, a serial I/O port for either multi-processor communications, I/O expansion or full duplex UART, and on-chip oscillator and clock circuits.

In addition, the 8XC52 has two software selectable modes of power reduction – idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

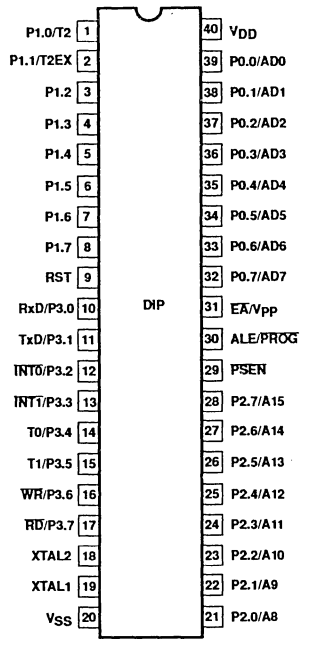
FEATURES

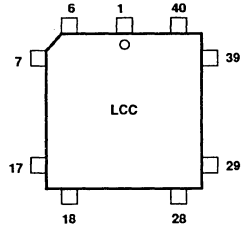
- 80C51 based architecture
- 8032/8052 compatible
  - 8k × 8 ROM (80C52)
  - 8k × 8 EPROM (87C52)
  - ROMless (80C32)
  - 256 × 8 RAM
  - Three 16-bit counter/timers
  - Full duplex serial channel
  - Boolean processor
- Memory addressing capability
  - 64k ROM and 64k RAM
- Power control modes:
  - Idle mode
  - Power-down mode
- CMOS and TTL compatible
- Two speed ranges:
  - 3.5 to 16MHz
  - 3.5 to 20MHz
- Five package styles
- Extended temperature ranges
- OTP package available

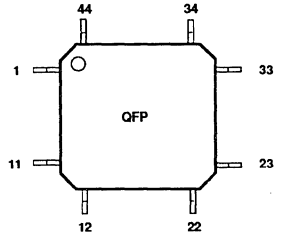
PART NUMBER SELECTION

ROMless	ROM	EPROM	TEMPERATURE °C AND PACKAGE	FREQ. (MHz)
P80C32EBP N	P80C52EBP N	P87C52EBP N	0 to +70, plastic DIP	16
P80C32EBA A	P80C52EBA A	P87C52EBA A	0 to +70, plastic LCC	16
		P87C52EBF FA	0 to +70, ceramic DIP	16
		P87C52EBL KA	0 to +70, ceramic LCC	16
P80C32EFP N	P80C52EFP N	P87C52EFP N	–40 to +85, plastic DIP	16
P80C32EFA A	P80C52EFA A	P87C52EFA A	–40 to +85, plastic LCC	16
		P87C52EFF FA	–40 to +85, ceramic DIP	16
		P87C52EFL KA	–40 to +85, ceramic LCC	16
P80C32EBB B	P80C52EBB B	P87C52EBB B	0 to +70, plastic QFP	16
P80C32EFB B	P80C52EFB B	P87C52EFB B	–40 to +85, plastic QFP	16
P80C32GBP N	P80C52GBP N	P87C52GBP N	0 to +70, plastic DIP	20
P80C32GBA A	P80C52GBA A	P87C52GBA A	0 to +70, plastic LCC	20
		P87C52GBF FA	0 to +70, ceramic DIP	20
		P87C52GBL KA	0 to +70, ceramic LCC	20
P80C32GFP N	P80C52GFP N	P87C52GFP N	–40 to +85, plastic DIP	20
P80C32GFA A	P80C52GFA A	P87C52GFA A	–40 to +85, plastic LCC	20
		P87C52GFF FA	–40 to +85, ceramic DIP	20
		P87C52GFL KA	–40 to +85, ceramic LCC	20
P80C32GBB B	P80C52GBB B	P87C52GBB B	0 to +70, plastic QFP	20
P80C32GFB B	P80C52GFB B	P87C52GFB B	–40 to +85, plastic QFP	20

PIN CONFIGURATIONS

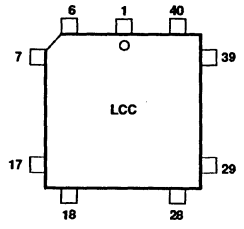






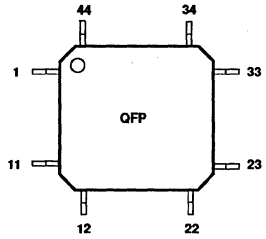
SEE NEXT PAGE FOR LCC AND QFP PIN FUNCTIONS.

LCC PIN FUNCTIONS



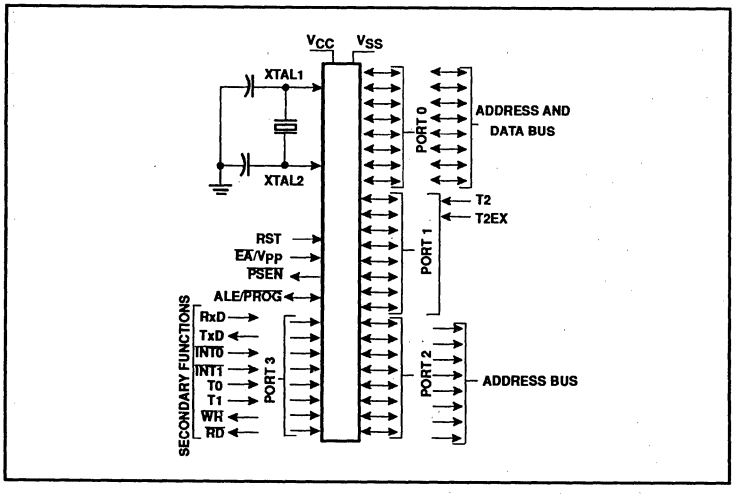
Pin	Function	Pin	Function
1	NC	23	NC
2	T2/P1.0	24	P2.0/A8
3	T2EX/P1.1	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6	30	P2.6/A14
9	P1.7	31	P2.7/A15
10	RST	32	$\overline{PSEN}$
11	RxD/P3.0	33	ALE/$\overline{PROG}$
12	NC	34	NC
13	TxD/P3.1	35	$\overline{EA}$/V_{PP}
14	$\overline{INT0}$/P3.2	36	P0.7/AD7
15	$\overline{INT1}$/P3.3	37	P0.6/AD6
16	T0/P3.4	38	P0.5/AD5
17	T1/P3.5	39	P0.4/AD4
18	$\overline{WR}$/P3.6	40	P0.3/AD3
19	$\overline{RD}$/P3.7	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	V_{SS}	44	V_{CC}

QFP PIN FUNCTIONS

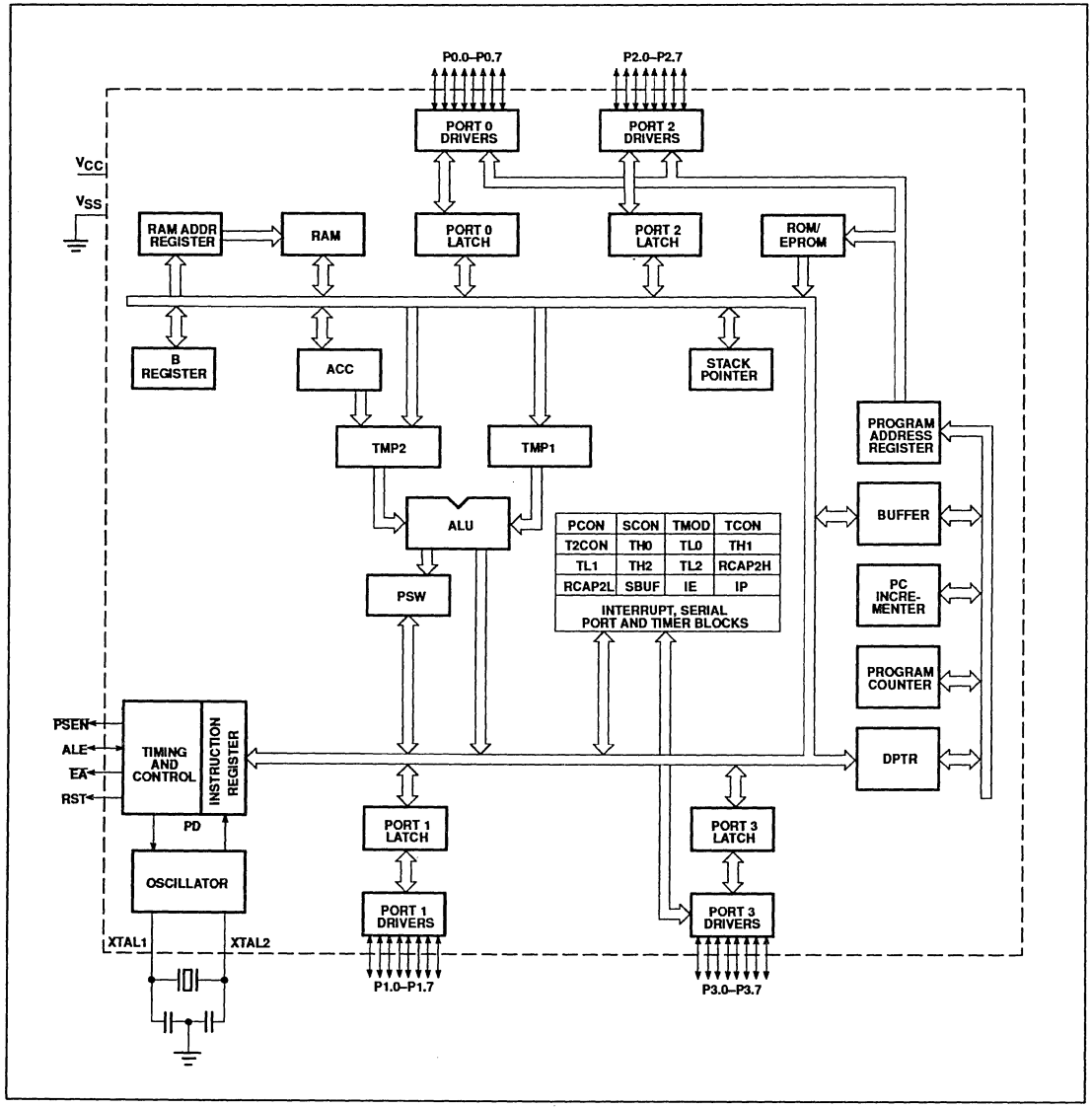


Pin	Function	Pin	Function
1	P1.5	23	P2.5/A13
2	P1.6	24	P2.6/A14
3	P1.7	25	P2.7/A15
4	RST	26	$\overline{PSEN}$
5	RxD/P3.0	27	ALE/$\overline{PROG}$
6	NC	28	NC
7	TxD/P3.1	29	$\overline{EA}$/V_{PP}
8	$\overline{INT0}$/P3.2	30	P0.7/AD7
9	$\overline{INT1}$/P3.3	31	P0.6/AD6
10	T0/P3.4	32	P0.5/AD5
11	T1/P3.5	33	P0.4/AD4
12	$\overline{WR}$/P3.6	34	P0.3/AD3
13	$\overline{RD}$/P3.7	35	P0.2/AD2
14	XTAL2	36	P0.1/AD1
15	XTAL1	37	P0.0/AD0
16	V_{SS}	38	V_{CC}
17	NC	39	NC
18	P2.0/A8	40	T2/P1.0
19	P2.1/A9	41	T2EXP/P1.1
20	P2.2/A10	42	P1.2
21	P2.3/A11	43	P1.3
22	P2.4/A12	44	P1.4

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{SS}	20	22	16	I	**Ground:** 0V reference.
V_{CC}	40	44	38	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–36	37–30	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 also outputs the code bytes during program verification in the 87C52. External pull-ups are required during program verification.
P1.0–P1.7	1–8	2–9	40–44 1–3	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Pins P1.0 and P1.1 also. Port 1 also receives the low-order address byte during program memory verification. Port 1 also serves alternate functions for timer 2:
	1	2	40	I	**T2 (P1.0):** Timer/counter 2 external count input.
	2	3	41	I	**T2EX (P1.1):** Timer/counter 2 trigger input.
P2.0–P2.7	21–28	24–31	18–25	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	5, 7–13	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 80C51 family, as listed below:
	10	11	5	I	**RxD (P3.0):** Serial input port
	11	13	7	O	**TxD (P3.1):** Serial output port
	12	14	8	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	9	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	10	I	**T0 (P3.4):** Timer 0 external input
	15	17	11	I	**T1 (P3.5):** Timer 1 external input
	16	18	12	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	13	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	4	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE/$\overline{PROG}$	30	33	27	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{PSEN}$	29	32	26	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$/V_{PP}	31	35	29	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 1FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 1FFFH. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	19	21	15	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	14	O	**Crystal 2:** Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the Logic Symbol, page 216.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-up reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-up, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON.

DESIGN CONSIDERATIONS

At power-on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

Table 1 shows the state of I/O ports during low current operating modes.

ROM CODE SUBMISSION

When submitting ROM code for the 80C52, the following must be specified:

1. 8k byte user ROM data
2. 32 byte ROM encryption key
3. ROM security bits.

ADDRESS	CONTENT	BIT(S)	COMMENT
0000H to 1FFFH	DATA	7:0	User ROM Data
2000H to 201FH	KEY	7:0	ROM Encryption Key FFH = no encryption
2020H	SEC	0	ROM Security Bit 1
2020H	SEC	1	ROM Security Bit 2 0 = enable security 1 = disable security

Security Bit 1: When programmed, this bit has two effects on masked ROM parts:

1. External MOVC is disabled, and
2. EA# is latched on Reset.

Security Bit 2: When programmed, this bit inhibits Verify User ROM.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Electrical Deviations from Commercial Specifications for Extended Temperature Range (87C52)
DC and AC parameters not included here are the same as in the commercial temperature range table.

DC ELECTRICAL CHARACTERISTICS

T_{amb} = –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNIT
			MIN	MAX	
V_{IL}	Input low voltage, except $\overline{EA}$		–0.5	$0.2V_{CC}$–0.15	V
V_{IL1}	Input low voltage to $\overline{EA}$		0	$0.2V_{CC}$–0.35	V
V_{IH}	Input high voltage, except XTAL1, RST		$0.2V_{CC}$+1	V_{CC}+0.5	V
V_{IH1}	Input high voltage to XTAL1, RST		$0.7V_{CC}$+0.1	V_{CC}+0.5	V
I_{IL}	Logical 0 input current, ports 1, 2, 3	V_{IN} = 0.45V		–75	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3	V_{IN} = 2.0V		–750	µA
I_{CC}	Power supply current: Active mode Idle mode Power-down mode	V_{CC} = 4.5–5.5V, Frequency range = 3.5 to 16MHz		 19 6 50	 mA mA µA

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Operating temperature under bias	0 to +70 or –40 to +85	°C
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ pin to V_{SS}	0 to +13.0	V
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Maximum I_{OL} per I/O pin	15	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.5	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (87C52)
T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V (80C32/80C52)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			MIN	TYP[1]	MAX	
V_{IL}	Input low voltage, except $\overline{EA}$[7]		–0.5		$0.2V_{CC}$–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$[7]		0		$0.2V_{CC}$–0.3	V
V_{IH}	Input high voltage, except XTAL1, RST[7]		$0.2V_{CC}$+0.9		V_{CC}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST[7]		$0.7V_{CC}$		V_{CC}+0.5	V
V_{OL}	Output low voltage, ports 1, 2, 3[9]	I_{OL} = 1.6mA[2]			0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$[9]	I_{OL} = 3.2mA[2]			0.45	V
V_{OH}	Output high voltage, ports 1, 2, 3, ALE, $\overline{PSEN}$[3]	I_{OH} = –60µA, I_{OH} = –25µA I_{OH} = –10µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
V_{OH1}	Output high voltage (port 0 in external bus mode)	I_{OH} = –800µA, I_{OH} = –300µA I_{OH} = –80µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
I_{IL}	Logical 0 input current, ports 1, 2, 3[7]	V_{IN} = 0.45V			–50	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3[7]	See note 4			–650	µA
I_{LI}	Input leakage current, port 0	V_{IN} = V_{IL} or V_{IH}			±10	µA
I_{CC}	Power supply current:[7] Active mode @ 12MHz[5] Idle mode @ 12MHz Power-down mode	See note 6		 11.5 1.3 3	 19 4 50	 mA mA µA
R_{RST}	Internal reset pull-down resistor		50		300	kohm
C_{IO}	Pin capacitance				10	pF

NOTES:

1. Typical ratings are not guaranteed. The values listed are at room temperature, 5V.
2. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. I_{OL} can exceed these conditions provided that no single output sinks more than 5mA and no more than two outputs exceed the test conditions.
3. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
4. Pins of ports 1, 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
5. I_{CC}MAX at other frequencies is given by: Active mode: I_{CC}MAX = 1.43 X FREQ + 1.9: Idle mode: I_{CC}MAX = 0.14 X FREQ +2.31, where FREQ is the external oscillator frequency in MHz. I_{CC}MAX is given in mA. See Figure 8.
6. See Figures 9 through 12 for I_{CC} test conditions.
7. These values apply only to T_{amb} = 0°C to +70°C. For T_{amb} = –40°C to +85°C, see table on previous page.
8. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
9. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 15mA (*NOTE: This is 85°C specification.)
   Maximum I_{OL} per 8-bit port: 26mA
   Maximum total I_{OL} for all outputs: 67mA
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

AC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or −40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (87C52)[1, 2, 3]

SYMBOL	FIGURE	PARAMETER	20MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	1	Oscillator frequency: **Speed Versions** 8XC52 E 8XC52 G			 3.5 3.5	 16 20	 MHz MHz
t_{LHLL}	1	ALE pulse width	60		$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	37		$t_{CLCL}-13$		ns
t_{LLAX}	1	Address hold after ALE low	30		$t_{CLCL}-20$		ns
t_{LLIV}	1	ALE low to valid instruction in		135		$4t_{CLCL}-65$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	37		$t_{CLCL}-13$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	130		$3t_{CLCL}-20$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		105		$3t_{CLCL}-45$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		40		$t_{CLCL}-10$	ns
t_{AVIV}	1	Address to valid instruction in		195		$5t_{CLCL}-55$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	200		$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	200		$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		160		$5t_{CLCL}-90$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		72		$2t_{CLCL}-28$	ns
t_{LLDV}	2, 3	ALE low to valid data in		250		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		285		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	100	200	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	125		$4t_{CLCL}-130$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	30		$t_{CLCL}-40$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	30		$t_{CLCL}-40$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	30	75	$t_{CLCL}-40$	$t_{CLCL}+25$	ns
External Clock							
t_{CHCX}	5	High time	20		20		ns
t_{CLCX}	5	Low time	20		20		ns
t_{CLCH}	5	Rise time		20		20	ns
t_{CHCL}	5	Fall time		20		20	ns
Shift Register							
t_{XLXL}	4	Serial port clock cycle time	600		$12t_{CLCL}$		ns
t_{QVXH}	4	Output data setup to clock rising edge	367		$10t_{CLCL}-133$		ns
t_{XHQX}	4	Output data hold after clock rising edge	20		$2t_{CLCL}-80$		ns
t_{XHDX}	4	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	4	Clock rising edge to input data valid		367		$10t_{CLCL}-133$	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. Interfacing the 80C32/52 to devices with float times up to 45ns is permitted. This limited bus contention will not cause damage to Port 0 drivers.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{\text{PSEN}}$
Q – Output data
R – $\overline{\text{RD}}$ signal
t – Time
V – Valid
W – $\overline{\text{WR}}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{\text{PSEN}}$ low.

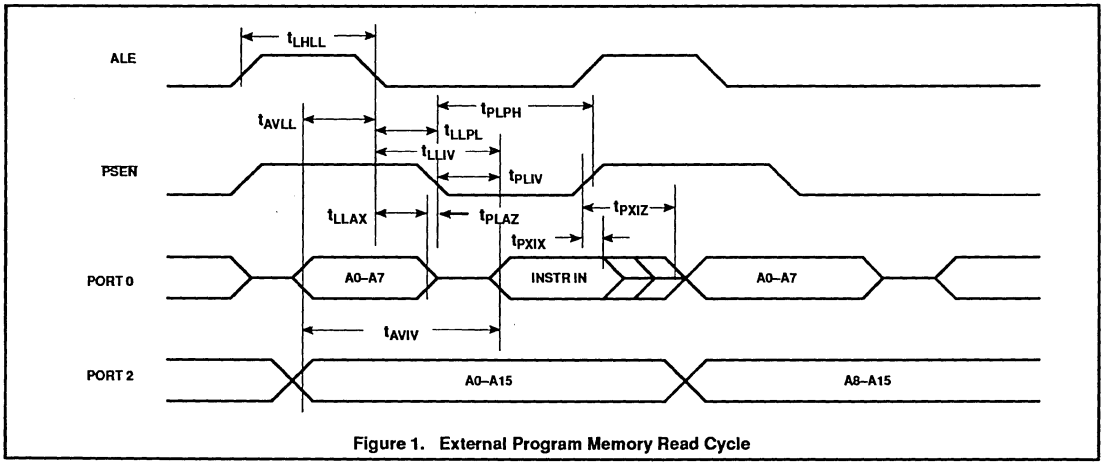


Figure 1. External Program Memory Read Cycle

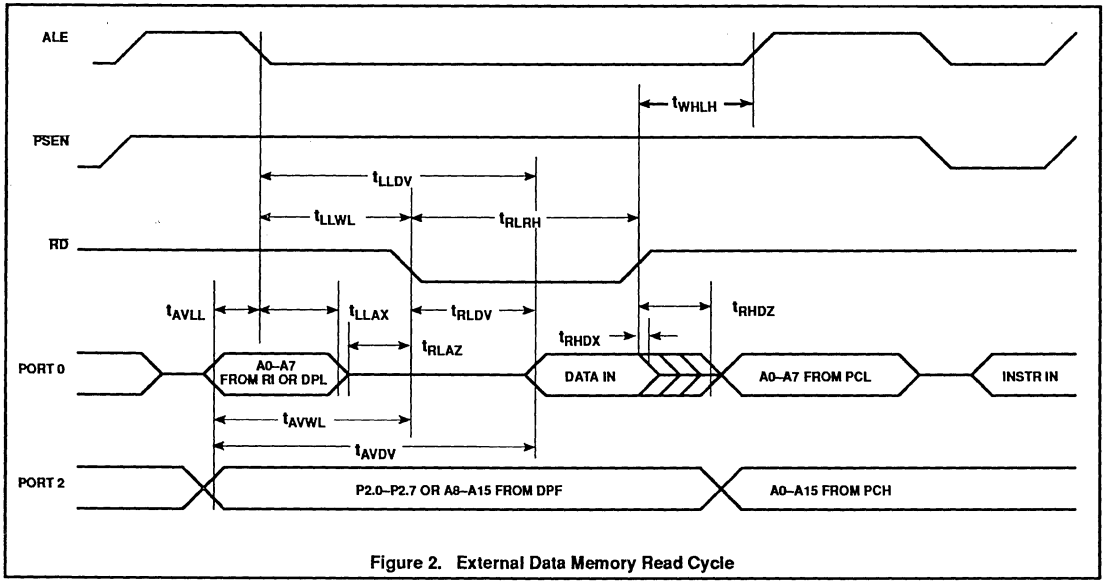


Figure 2. External Data Memory Read Cycle

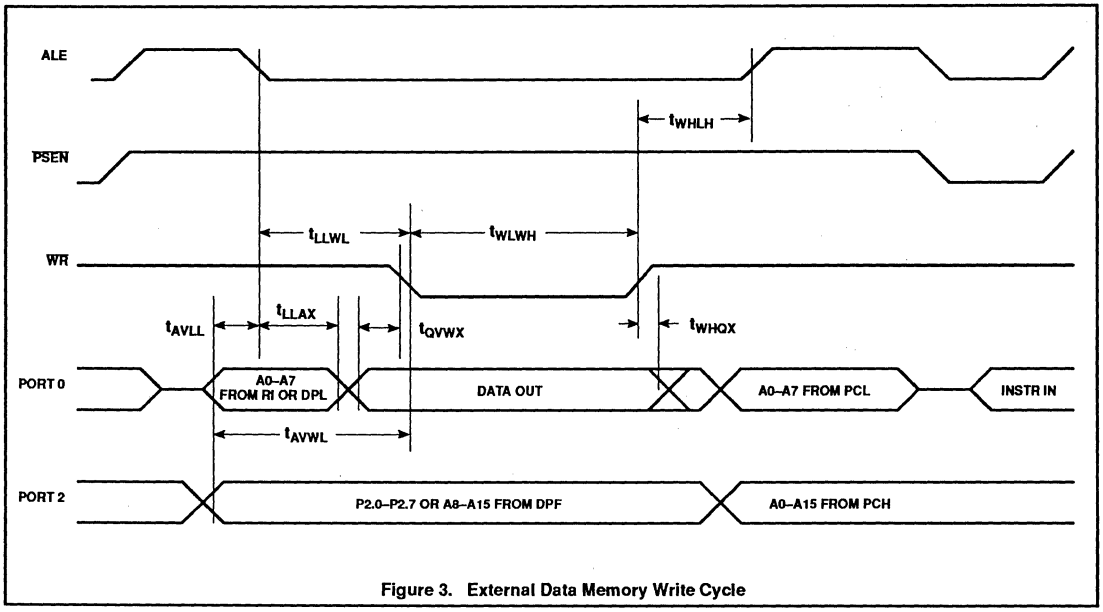


Figure 3. External Data Memory Write Cycle

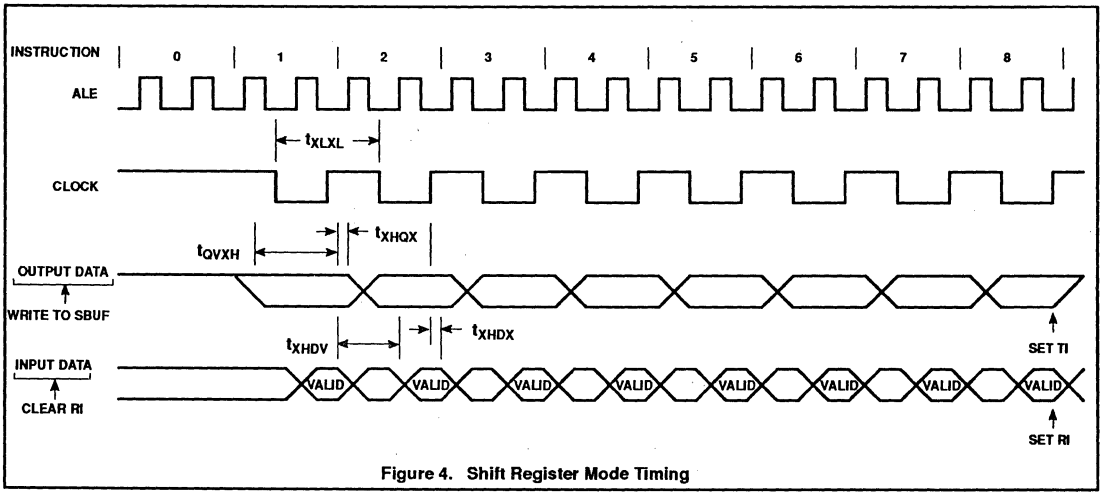


Figure 4. Shift Register Mode Timing

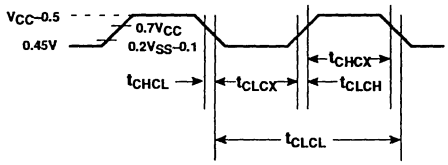
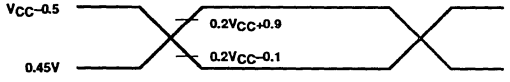
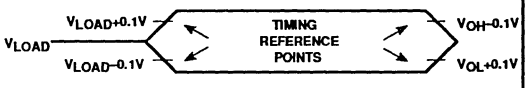


Figure 5. External Clock Drive



NOTE:
AC inputs during testing are driven at V_{CC} –0.5 for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at V_{IH} min for a logic '1' and V_{IL} for a logic '0'.

Figure 6. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 7. Float Waveform

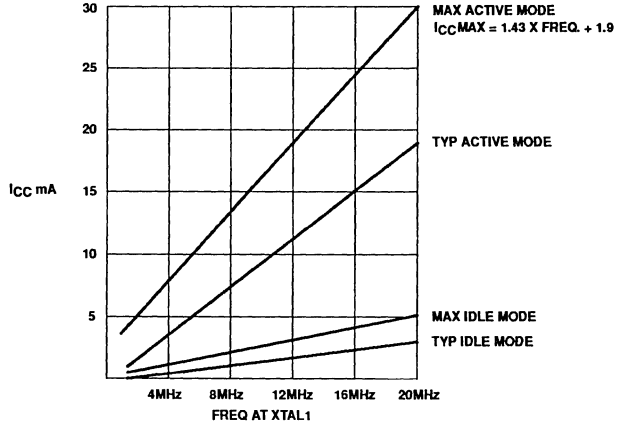


Figure 8. I_{CC} vs. FREQ
Valid only within frequency specifications of the device under test

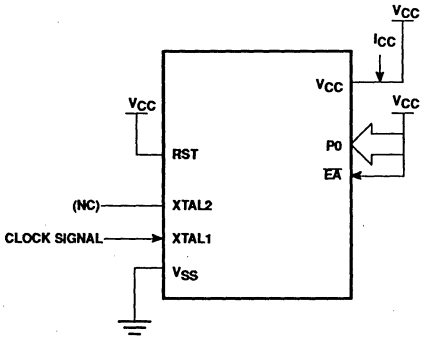


Figure 9. I_{CC} Test Condition, Active Mode
All other pins are disconnected

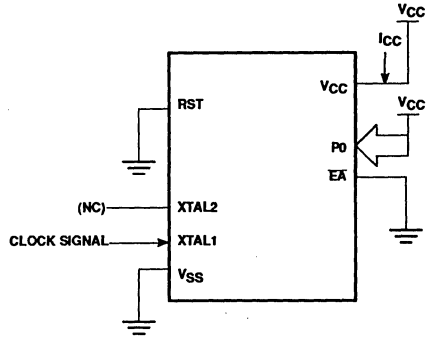


Figure 10. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

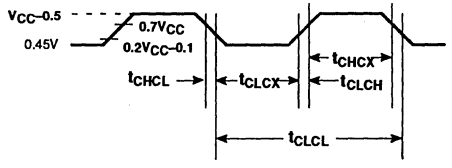


Figure 11. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

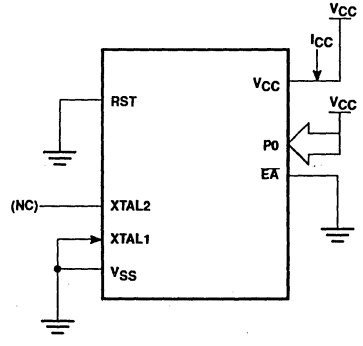


Figure 12. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

EPROM CHARACTERISTICS

The 87C52 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C52 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C52 manufactured by Philips.

Table 2 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the secuity bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 13 and 14. Figure 15 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 13. Note that the 87C52 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 13. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 2 and 3 specified in Table 2 are held at the 'Program Code Data' levels indicated in Table 2. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 14.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 1FH, using the 'Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the security bits, repeat the 25 pulse programming sequence using the 'Pgm Security Bit' levels. After one security bit is programmed, further programming of the code memory and encryption table is disabled. However, the other security bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If security bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 1 and 2 as shown in Figure 15. The other pins are held at the 'Verify Code Data' levels indicated in Table 2. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(031H) = 97H indicates 87C52

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 2, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-s/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000μW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 2. EPROM Progamming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm security bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm security bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:

1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.

* ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100μs (±10μs) and high for a minimum of 10μs.

™Trademark phrase of Intel Corporation.

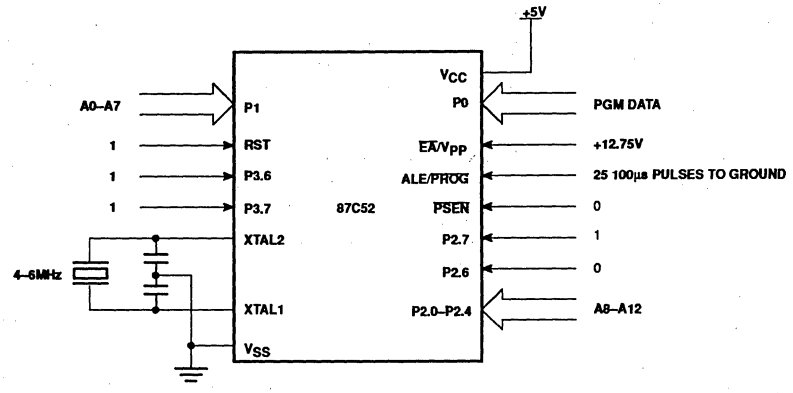


Figure 13. Programming Configuration

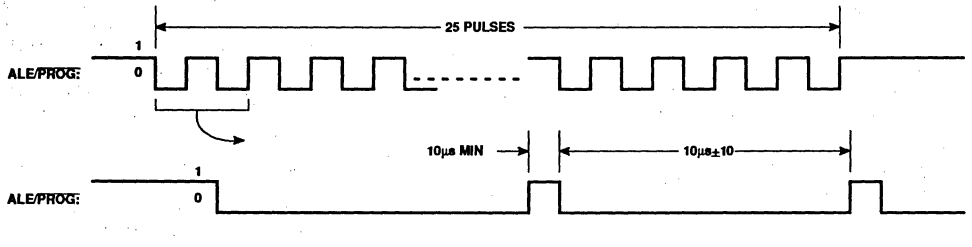


Figure 14. PROG Waveform

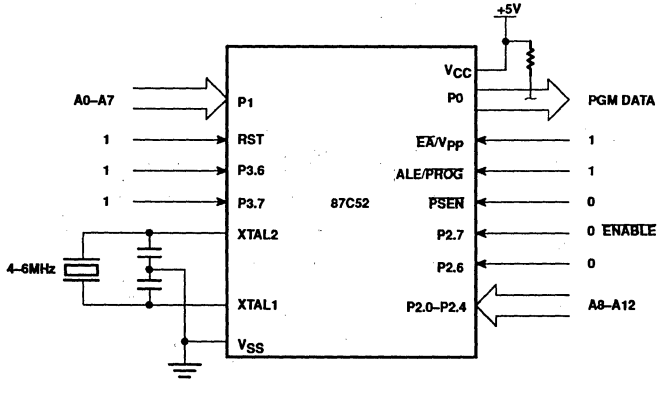


Figure 15. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_{amb} = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 16)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

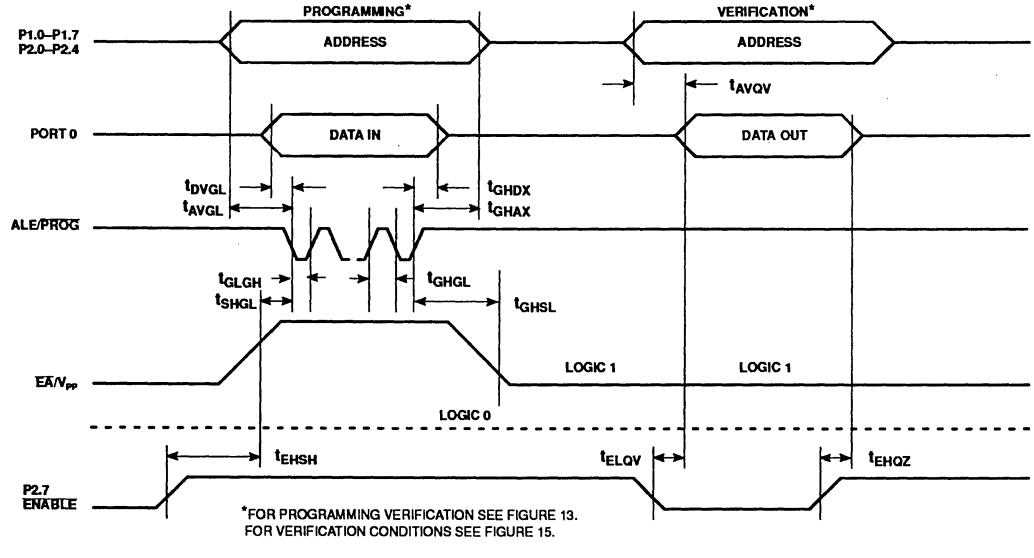


Figure 16. EPROM Programming and Verification

8XC053/54 OVERVIEW

The 8XC053/54 is an 80C51 derivative microcontroller that is designed to control and display text information on raster scanned video displays, such as televisions and video monitors.

The part consists of:

- An 80C51 microcontroller core
- An On-Screen Display (OSD) function block
- Pulse-Width Modulators and an Analog-to-Digital converter
- High voltage (12V) and LED drive outputs (10mA)

The part is available with either 8k (83C053) or 16k (83C054) bytes of ROM Program Memory, or with 16k bytes of EPROM Program Memory. The parts are functionally identical except for the Program Memory differences and are collectively referred to as the 8XC053/54.

The EPROM version, the 87C054, is used for product development and for initial and lower volume production quantities. Note that, owing to requirements for I/O pins, the parts are not designed to execute programs from external memory.

The parts are available in a 42-pin Plastic Shrink Dual In-Line package as ROM (83C053 and 83C054) and One-Time-Programmable versions (87C054).

Development systems and EPROM Programmers for the product are available from several sources (see section on Development Systems and EPROM Programmers).

The basic features of the 8XC053/54 are:

- 80C51 based architecture
- 192 bytes of on-chip RAM
- OSD functional block (described in detail later)
  - Inputs to the OSD are:
    Horizontal Sync (HSYNC), Vertical Sync (VSYNC), and a Dot, or Pixel, clock from an external oscillator (locked to HSYNC).
  - Outputs from the OSD are:
    Three digital video outputs (RGB), a Video Multiplexing signal to select between 8XC053/54 and base video and a Foreground/Background output to control overlay background.
- A Character Generator Memory with 60 character bit-maps, each 18 (vertical) × 14 (horizontal).
- 128 × 10 Display RAM. This memory is written into by the 80C51 CPU and read by the OSD to fetch the pixel data to display from the Character Generator Memory.
- Eight 6-bit Pulse Width Modulators (PWM) and one 14-bit high-precision PWM.
- 4-bit D/A converter and comparator with a 3-input multiplexer allowing implementation of an A/D in software.
- Four high-current (10mA) open drain outputs.
- Twelve high voltage (+12V) open drain outputs.

A typical application of the 8XC053/54 would be as the central microcontroller in a television set of video monitor. The microcontroller would perform the following functions:

- Interface with the Remote control and keypad, receive and carry out the user's commands.
- Perform the OSD function to display all available control options and settings of user-controllable functions such as channel selection, brightness, volume, hue, tint, frequency settings (for multi-sync monitors), etc.
- Generate, via the on-chip PWMs, control voltages for the user-controllable functions to execute user commands.
- Perform test and diagnostic functions.

For a standard NTSC TV signal with an HSYNC frequency of 15,750kHz and a VSYNC of nominally 60Hz, there are roughly 50 microseconds of active horizontal scan line available. The maximum pixel clock frequency is 8MHz, and therefore roughly 400 pixels of resolution can be obtained. At 14 dots per character, this means 28 character times per horizontal scan. If the 12 dot per character display mode is used, that means 33 character times per horizontal scan. Allowing for edge effects, 26 characters (14 across) or 31 characters (12 across) can be displayed.

Note that VGA rates and higher can be used. The minimum character dot size will be a function of the VGA frequency used. For a 640 × 480 display, running at 33kHz, the equivalent 8XC053/54 pixel resolution is about 160 across (because of the 8MHz clock and allowing for overscan). This means that status and diagnostic information can be displayed on video monitors.

The 8XC053/54 On-Screen Display (OSD) Block

Figure 1 is a conceptual drawing of the OSD block on the 8XC053/54.

It shows the CPU writing into the 128 × 10 display RAM, which is dual-ported to allow the CPU to write into it at any time, including when it is being read out by the OSD logic. The 10-bit wide data coming out of the display RAM is used to access the appropriate character in the Character Generator memory (6-bits) and to specify character and display control functions (4-bits). Timing for the OSD is controlled by the HSYNC, VSYNC, and dot clock inputs.

The 8XC053/54 features an advanced OSD function with some unique features:

1. User-definable display format: The OSD does not restrict the user to a fixed number of lines with a fixed number of characters per line as other competing alternatives do.

   Using a fixed number of lines restricts the generation of displays that can be differentiated from others that use the same chip and places limits on screen content.

   Using a fixed number of characters per line wastes display RAM if a line has less than the full number of displayable characters (it has to be padded with non-visible characters).

   The OSD on the 8XC053/54 defines a control character (New Line) that has the same function as a Carriage Return and Line Feed. When the OSD circuitry fetches this character from display RAM it stops displaying further characters, waits for the next horizontal scan line, and starts displaying the next character in display RAM after the New Line character was received. The number of lines is thus up to the user, within the limits of the display and memory, as are the number of characters per line. This allows far better control of the appearance of the On Screen Display.

2. Dual-Ported Display RAM: The OSD has a true display RAM instead of a character line buffer. This display RAM is dual ported to allow updating the display RAM at any time instead of having to wait for a vertical retrace. Vertical Sync interrupts are supported if flicker-free updates are required.

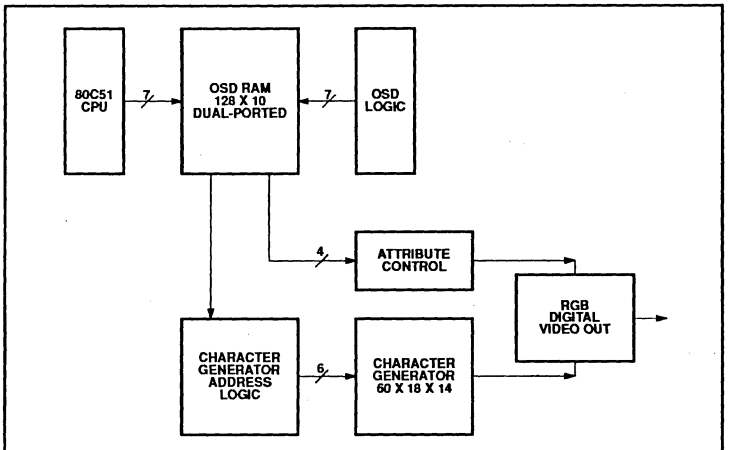


Figure 1. On-Screen Display (OSD) Block

3. Colors selectable by character: Characters can be displayed on a background of the base video or a programmable background color. The background color is selectable by word and the choice of background (base video/user programmed color) by character.

4. Programmable character size: Normal characters are displayed as 18 × 14 bit-maps. In an interlaced display, 2 fields are displayed so that one actually sees a 36 × 14 pixel size character. The part has a double height and width mode which displays 36 × 28 pixel size bit maps per field. For use in non-interlaced systems, the part has a double height mode so that the displayed characters have the same pixel size (36 × 14) as on an interlaced display.

5. Character shadowing: When characters are displayed overlaid on a background of base video, a black border around the characters makes them highly legible. This feature is called shadowing. The 8XC053/54 has 8 shadowing modes to allow the user to select various partial shadow modes as well as full surround shadow.

6. Short Rows: This mode only displays 4 horizontal lines. it is used for generating underlines.

7. Programmable polarities: Inputs to the OSD can be programmed to be recognized as Active-LOW or Active-HIGH, also the outputs from the OSD. Coupled with the 12V outputs, this allows direct interfacing to most video signal processing circuits.

8. Programmable horizontal and vertical positions: A pair of registers allow defining the starting point of the display.

9. HSYNC locked dot-clock oscillator: The chip is designed to use an L-C oscillator circuit that is started at the trailing edge of HSYNC and stopped at its leading edge. In practice, this gives a highly consistent delay from HSYNC to oscillator start and is stable from scan line to scan line so that no left margin effects are seen.

10. Character Generator memory in EPROM: On the 87C054, the Character Generator memory is in EPROM. This feature allows quick and inexpensive font development and refinement against the alternative of doing a masked ROM version to see how the final fonts will appear.

Differences from the 80C51

Memory Organization
The 8XC053/54 differs from the 80C51 in that it has either 8k or 16k or on-chip program memory, and it is not externally expandable. The size of the on-chip data RAM also differs in that it is 192 bytes.

The display RAM, where the contents of the screen to be displayed are wrtten by the 80C51 CPU, and read out and displayed by the OSD, is 128 deep by 10 bits wide. The 10 bits are composed of 6 address bits and 4 attributs bits; the 6 address bits form the address of the character in the 60 × 18 × 14 bit-map Character Generator memory, the 4 bits are used to control the attributes of that character and of the display.

Special Function Registers (SFRs)
The 8XC053/54 contains 17 additional SFRs in addition to those found on the 80C51. Six of the additional registers control the OSD block, ten control the PWMs, and one controls the D/A and voltage comparator.

The six registers that control the OSD block are:

OSAD: Specifies the 7-bit address of the display RAM that the CPU uses to write into.

OSAT: Contains the 4 attribute bits for character and display control.

OSDT: Contains the 6-bit address of the character in the 60 × 18 × 14 Character Generator bit-map memory.

Writing into OSDT causes the contents of OSDT to be concatenated with the contents of OSAT to form a 10-bit word that is then written into the 128 × 10 display RAM at the address pointed at by OSAD. OSAD is auto-incremented after the write. Thus, for a series of sequential writes, the CPU does not need to increment OSAD. If the attribute values do not change, this can be left unchanged too.

OSCON: Contains the VSYNC interrupt flag, programmable polarity bits, double height and background flag output control.

OSORG: Contains the horizontal and vertical starting positions for the display in terms of multiples of horizontal and vertical scan lines.

OSMOD: Contains bits specifying 12/14 column display and shadowing modes. Also has bits contrlling OSD enabling/disabling.

The ten SFRs that control the PWMs contain values and enable bits for the 8 6-bit PWMs (PWM0–PWM7) and values and enable bits for the 14-bit PWM (TDACL and TDACH).

The D/A function is controlled by the SAD register which allows a successive approximations A/D function to be performed in software.

Reduced Power Modes
There is no Idle Mode in the 8XC053/54. Power Down is supported, but because of a resistor ladder in the A/D, the power-down current is only reduced to 5mA (over V_{CC} from 2 to 6V).

Interrupts
To enable flick-free updating of the display, a VSYNC interrupt is implemented that can be used by the DPU to update the display RAM during the vertical retrace. Since the part does not have a UART, that interrupt is removed. Also, all interrupts have equal priority on this part, so there is no IP register. Note that to facilitate Pulse-Width measurement, one of the external interrupts (External Int 1) will generate an interrupt, if enabled, on both edges to allow easy software pluse width measurement.

Table 1. 8XC053/54 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt enable	A8H	EA	–	–	EVS	ET1	EX1	ET0	EX0	0x000000B
OSAD	On-screen address	9A									
OSAT	On-screen attributes	98									
OSDT	On-screen data	99									
OSCON	On-screen display control	C0									
OSMOD	On-screen display mode	C1									
OSORG	On-screen display origin	C2									

Table 1. 8XC053/54 Special Function Registers (Continued)

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB LSB								RESET VALUE
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	PWM7	PWM6	PWM5	PWM4	PWM3	PWM2	PWM1	TDAC	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	–	–	–	–	PWM0	ADI2	ADI1	ADI0	FFH
P2*	Port 2	A0H	A7	A6	A5	A4	A3	A2	A1	A0	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	–	–	–	–	$\overline{INT0}$	$\overline{T0}$	$\overline{INT1}$	–	FFH
PCON	Power control	87H	–	–	–	–	GF1	GF0	PD		0xxxxxxxB
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
PWM0	Lo-res pulse width modulators	D4									
PWM1	Lo-res pulse width modulators	D5									
PWM2	Lo-res pulse width modulators	D6									
PWM3	Lo-res pulse width modulators	D7									
PWM4	Lo-res pulse width modulators	DC									
PWM5	Lo-res pulse width modulators	DD									
PWM6	Lo-res pulse width modulators	DE									
PWM7	Lo-res pulse width modulators	DF									
SAD	D/A and voltage comparator	D8									
SP	Stack pointer	81H									07H
TDACH	Hi-res pulse width modulators	D3									
TDACL	Hi-res pulse width modulators	D2									
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
TH0	Timer high 0	8CH									00H
TH1	Timer high 1	8DH									00H
TL0	Timer low 0	8AH									00H
TL1	Timer low 1	8BH									00H
TMOD	Timer mode	89H	–	–	–	T1M	–	CT0	–	T0M	00H

* Bit addressable

Microcontroller for television and video (MTV)

83C053/83C054/87C054

DESCRIPTION

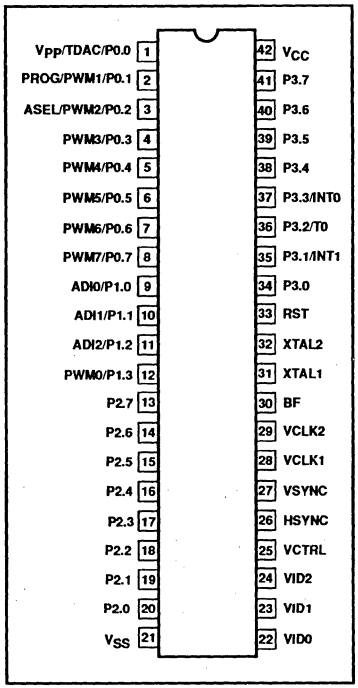
The Microcontroller for Television and Video (MTV) applications is a derivative of Philips' industry-standard 80C51 microcontroller that is intended for use as the central control mechanism in a television receiver or tuner. Providing tuner functions and an On Screen Display facility, it represents a next-generation replacement for the currently available parts.

The MTV is available in either an 8K masked ROM, 16K masked ROM, or 16K One Time Programmable (OTP) EPROM version. The only difference between these versions is the size or type of program memory.

FEATURES

- 8192 × 8 masked ROM (83C053), 16384 × 8 masked ROM (83C054), or 16384 × 8 OTP EPROM (87C054)
- 192 × 8 RAM
- On Screen Display (OSD) Controller
- Three digital video outputs
- Multiplexer/mixer and background intensity controls
- Flexible formatting with OSD New Line Option
- 128 × 10 display RAM
- 60 × 18 × 14 character generator ROM
- Eight text-shadowing modes
- Text color selectable per character
- Background color selectable per word
- Background color vs. video selectable per character
- Eight 6-bit pulse width modulators for analog voltage integration
- One 14-bit PWM for high-precision voltage integration
- D/A converter and comparator with three-input multiplexer
- Nine dedicated I/Os plus 28 port bits
- 15 port bits have alternate uses
- Four high-current open-drain port outputs
- 12 high-voltage (+12V) open drain outputs
- Programmable video input and output polarities
- 80C51 instruction set
- No external memory capability
- 42-pin shrunk DIP (0.07-inch center pins)
- High-speed CMOS technology
- 5V ± 10% operation

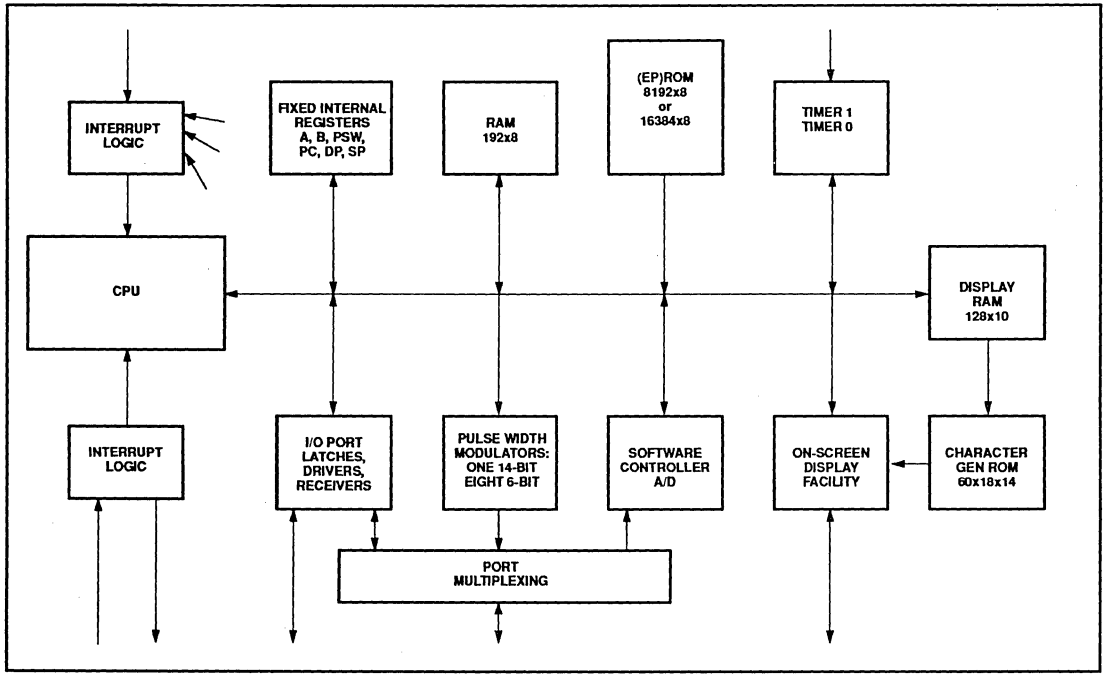
PIN CONFIGURATION



PART NUMBER SELECTION

ROM	EPROM	TEMPERATURE AND PACKAGE	FREQUENCY
P83C053BBP NB		0 to +70°C, plastic DIP	3.5 to 12MHz
P83C054BBP NB	P87C054BBP NB	0 to +70°C, plastic DIP	3.5 to 12MHz

BLOCK DIAGRAM



PIN DESCRIPTIONS

MNEMONIC	PIN NO. DIP	TYPE	NAME AND FUNCTION
VCLK1	28	I	**Video Clock 1:** Input for the horizontal timing reference for the On Screen Display facility. VCLK1 and VCLK2 are intended to be used with an external LC circuit to provide an on-chip oscillator. The period of the video clock is determined such that the width of a pixel in the On Screen Display is equal to the inter-line separation of the raster.
VCLK2	29	O	**Video Clock 2:** Output from the on-chip video oscillator.
HSYNC	26	I	**Horizontal Sync:** A dedicated input for a TTL-level version of the horizontal sync pulse. The polarity of this pulse is programmable; its trailing edge is used by the On Screen Display facility as the reference for horizontal positioning.
VSYNC	27	I	**Vertical Sync:** A dedicated input for a TTL-level version of the vertical sync pulse. The polarity of this pulse is programmable, and either edge can serve as the reference for vertical timing.
VID2:0	22–24	O	**Digital Video bus:** Three totem pole outputs comprising digital RGB (or other color encoding) from the On Screen Display facility. The polarity of these outputs is controlled by a programmable register bit.
VCTRL	25	O	**Video Control:** A totem-pole output indicating whether the On Screen Display facility is currently presenting active video on the VID2:0 outputs. This signal should be used to control an external multiplexer (mixer) between normal video and the video derived from VID2:0. The polarity of this outputs is controlled by a programmable register bit.
BF	30	O	**Background/Foreground:** A totem-pole output which, when VCTRL is active, indicates whether the current video data represents a foreground (low) or background (high) dot in a character. This signal can be used to reduce the intensity of the background color and thus emphasize the text. If a 40-pin version of this part is ever produced, BF will not be pinned out.
P0.0-P0.7	1–8	I/O	**Port 0:** An 8-bit open-drain bidirectional port. Port 0 pins that have ones written to them float, and in that state can be used as high-impedance inputs. The port 0 pins can also serve as outputs from the high-precision Pulse Width Modulator (TDAC) and seven of the eight lower-precision Pulse Width Modulator functions. For each PWM block, a register bit controls whether the corresponding pin is controlled by the block or by port 0; port 0 controls the pin immediately after a Reset. Regardless of how each pin is controlled, it can be externally pulled up as high as +12V±5%, and the state of the pin can be read from the Port 0 register by the program.
	1	I	**V_{PP} (P0.0)** – This pin receives the 12V programming supply voltage during EPROM programming.
	2	I	**PROG (P0.1)** – This pin receives the programming pulses during EPROM programming.
	3	I	**ASEL (P0.2)** – Input which indicates which bits of the EPROM address are applied to port 2.
	1	O	**TDAC (P0.0)** – This is the output for the 14-bit high-precision PWM.
	2–8	O	**PWM1–7 (P0.1–P0.7)** – Outputs for the 6-bit PWMs 1 through 7.
P1.0-P1.3	9–12	I/O	**Port 1:** A 4-bit open-drain bidirection port. Port 1 pins that have ones written to them float, and in that state can be used as high-impedance inputs. P1.3 can also serve as the eighth lower-precision Pulse Width Modulator output (PWM0), and can be externally pulled up as high as +12V±5%. P1.2:0 have optional alternate use as ADI2:0, inputs to the Software A/D conversion facility. If a 40-pin version of this part is ever produced, P1.3/PWM0 will not be pinned out. Any of the port 1 pins are driven low if the corresponding port register bit is written as 0, or, for P1.3 only, if the TDAC module presents a 0. The state of the pin can always be read from the port register by the program.
	9–11	I	**ADI0–2 (P1.0–P1.2)** – Inputs for the software A/D facility.
	12	O	**PWM0 (P1.3)** – Output for the PWM0 6-bit PWM.
P2.0-P2.7	20–13	I/O	**Port 2:** An 8-bit open-drain bidirectional port. Port 2 pins that have ones written to them float, and in that state can be used as high-impedance inputs. P2.3:0 have high current capability (10 mA at 0.5V) for LEDs. Any of the port 2 pins are driven low if the port register bit is written as 0. The state of the pin can always be read from the port register by the program.
P3.0-P3.7	34–42	I/O	**Port 3:** An 8-bit open-drain bidirectional port. Port 3 pins that have ones written to them float, and in that state can be used as high-impedance inputs. P3.0, P3.4, and P3.7 can be externally pulled up as high as +12V±5%, while P3.5 and P3.6 have 10mA drive capability. Some of the port 3 pins can also serve alternate functions, as follows:
	35	I	**INT1 (P3.1)** – External Interrupt 1.
	36	I	**T0 (P3.2)** – Timer 0 external input.
	37	I	**INT0 (P3.3)** – External Interrupt 0.

PIN DESCRIPTIONS (Continued)

MNEMONIC	PIN NO. DIP	TYPE	NAME AND FUNCTION
RST	33	I	**Reset:** If this pin is high for two machine cycles (24 oscillator periods) while the oscillator is running, the MTV is reset. Also, this pin is used as a serial input to enter a test or EPROM programming mode, as on the 87C751.
XTAL1	31	I	**Crystal 1:** Input to the inverting oscillator amplifier and clock generator circuit that provides the timing reference for all MTV logic other than the OSD facility. XTAL1 and XTAL2 can be used with a quartz crystal or ceramic resonator to provide an on-chip oscillator. Alternatively, XTAL1 can be conncected to an external clock, and XTAL2 left unconnected.
XTAL2	32	O	**Crystal 2:** Output from the inverting oscillator amplifier.
V_{CC}		I	**Power Supply:** This is the power supply for normal and power-down modes.
V_{SS}		I	**Ground:** 0V reference.

ROM CODE SUBMISSION

When submitting a ROM code for the 83C053 or 83C054, the following must be specified:

1. The 8k byte (83C053), or 16k byte (83C054) user ROM program.
2. The OSD ROM space.

This information can be submitted in an 87C054, or in two EPROMs (2764), or electronically on the ROM Code Bulletin Board (see your local sales office for the number).

ROM CODE SUBMITTAL REQUIREMENTS

ADDRESS	CONTENT	COMMENT
0000H to 1FFFH (83C053) 000H to 3FFFH (83C054)	DATA	User ROM data
C000H to CFFFH	OSD	On-Screen Display character table

PROGRAMMING THE OSD EPROM

Overview

The OSD EPROM space starts at location C000H and ends at location CFFFH. However, not all locations within this space are used, due to the addressing scheme of the OSD.

The start location of the next character can be calculated by adding 40H to the start location of the previous character. For example, character #1 starts at C000H; then characters 2, 3, and 4 start at C040H, C080H, and C0C0H, respectively.

Character Description

Each character is 14 bits wide by 18 lines high.

A character is split about a vertical axis into two sections, UPPER and LOWER. Each section contains 7 bits of the character, such that the LOWER section contains 1–7 and the UPPER section contains bits 8–14.

NOTE: During programming and verification, each section is programmed using bytes of DATA. The MSB of the DATA is not used; however, the MSB location physically exists, and so will program and verify.

The LOWER section of the character is programmed when the LSB of the program address equals 0, and the UPPER section when the LSB equals 1.

Character Programming

An example of an OSD character bit map, and the program DATA to obtain that character is shown in Table 1.

OSD EPROM Bit Map

The mapping for the full OSD EPROM is shown in Table 2.

Example

To program the character given above into the first character location of the OSD EPROM would require the following address/DATA sequence:

C000/00H;	C001/00H;	C002/00H;
C003/00H;	C004/0CH;	C005/1EH;
C006/0CH;	C007/1EH;	C008/0CH;
C009/1EH;	C00A/0CH;	C00B/1EH;
C00C/0CH;	C00D/1EH;	C00E/0CH;
C00F/1EH;	C010/7CH;	C011/1FH;
C012/7CH;	C013/1FH;	C014/7CH;
C015/1FH;	C016/0CH;	C017/1EH;
C018/0CH;	C019/1EH;	C01A/0CH;
C01B/1EH;	C01C/0CH;	C01D/1EH;
C01E/0CH;	C01F/1EH;	C020/00H;
C021/00H;	C022/00H;	C023/00H

Table 1. Example of an OSD Character Bit Map

CHARACTER BIT MAP		PROGRAM DATA	
	UPPER LOWER ←— —→←— —→ 11111 43210987654321	UPPER	LOWER
Line 1 →	00000000000000	X0000000	X0000000
Line 2 →	00000000000000	X0000000	X0000000
Line 3 →	00111100001100	X0011110	X0001100
Line 4 →	00111100001100	X0011110	X0001100
Line 5 →	00111100001100	X0011110	X0001100
Line 6 →	00111100001100	X0011110	X0001100
Line 7 →	00111100001100	X0011110	X0001100
Line 8 →	00111100001100	X0011110	X0001100
Line 9 →	00111100001100	X0011110	X0001100
Line 10 →	00111100001100	X0011110	X0001100
Line 11 →	00111111111100	X0011111	X1111101
Line 12 →	00111111111100	X0011111	X1111101
Line 13 →	00111111111100	X0011111	X1111101
Line 14 →	00111111111100	X0011111	X1111101
Line 15 →	00111100001100	X0011110	X0001100
Line 16 →	00111100001100	X0011110	X0001100
Line 17 →	00111100001100	X0011110	X0001100
Line 18 →	00111100001100	X0011110	X0001100
	00111100001100	X0011110	X0001100
	00111100001100	X0011110	X0001100
	00000000000000	X0000000	X0000000
	00000000000000	X0000000	X0000000

NOTE:
X can be 0 or 1, and will program and verify correctly.

Table 2. OSD EPROM Bit Map

CHARACTER NO.	ADDRESS	CHARACTER LINE NO.	COMMENTS
1	C000 C001 C002 C003 • • • C022 C023 C024–C03F	1 1 2 2 • • • 18 18 Unused	Lower byte Upper byte Lower byte Upper byte • • • Lower byte Upper byte
2	C040–C063 C064–C07F	1–18 Unused	
3	C080–C0A3 C0A4–C0BF	1–18 Unused	
• • •	• • •	• • •	• • •
62	CFC0–CFE3 CFE4–CFFF	1–18 Unused	NEWLINE
63	CFC0–CFE3 CFE4–CFFF	1–18 Unused	BSPACE
64	CFC0–CFE3 CFE4–CFFF	1–18 Unused	SPLITBSPACE

NOTE:
Locations 62, 63, and 64 should be programmed to 0's.

COMPARISON TO THE 80C51

The elements of the MTV are shown in the Block Diagram. The features of the MTV are identical to those of the 80C51, except as noted herein.

Pinout and Testing

Since neither data nor program memory is externally expandable on the MTV, the 80C51 pins ALE, $\overline{EA}$, and $\overline{PSEN}$ are not implemented on the MTV.

I/O Ports

On both the 80C51 and the MTV, port 0 is open-drain, but on the 80C51 it can be used for external memory expansion while on the MTV its alternate use is for Pulse Width Modulated outputs.

On the 80C51, port 1 is 8 bits, is mostly unallocated (general purpose), and is quasi-bidirectional (that is, having a weak pullup transistor that can be overdriven). On the MTV it is a 4-bit open-drain port, and includes alternate uses for analog inputs and a PWM output.

On the 80C51, port 2 is quasi-bidirectional and can be used for external memory expansion; on the MTV, port 2 is open-drain and unallocated.

On the 80C51, port 3 is quasi-bidirectional and all eight bits have alternate uses. On the MTV, three port 3 bits have some of the same alternate uses as on the 80C51 but not necessarily on the same pins, while five pins are open-drain and unallocated.

Idle Mode

The idle mode is not implemented on the MTV.

Power-Down Mode

The power-down mode is not implemented on the MTV. The PCON register has the following format:

PCON

7	6	5	4	3	2	1	0
–	–	–	–	GF1	GF0	–	–

Interrupts

The interrupt facilities of the MTV differ from those of the 80C51 as follows:

1. Since there is not a serial port, there are no interrupts nor control bits relating to this interrupt. The interrupts and their vector addresses are as follows:

Event	Program Memory Address
Reset	000
External INT0	003
Timer 0	00B
External INT1	013
Timer 1	01B
VSync Start	023

2. The VSYNC input used by the On Screen Display facility can generate an interrupt. The active polarity of the pulse is programmable, as described in a later section. The interrupt occurs at the leading edge of the pulse.
3. External Interrupt 1 is modified so that an interrupt is generated when the input switches in either direction (on the 8051, there is a programmable choice between interrupt on a negative edge or a low level on INT1). This facility allows for software pulse-width measurement handling of a remote control.
4. The IP register is not used, and the IE register is similar to that on the 80C51:

IE

7	6	5	4	3	2	1	0
EA	–	–	EVS	ET1	EX1	ET0	EX0

Six-Bit PWM DACs

The structure of these modules is shown in Figure 2. First, the basic MCU clock is divided by 4 to get a waveform that clocks a 6-bit counter which is common to all the PWMs, including the 14-bit one. This divided clock is hereafter called the PWM counter clock.

Each PWM block has a special function register PWMn arranged as follows:

PWM0-PWM7

7	6	5	4	3	2	1	0
PWE	–	PV5	PV4	PV3	PV2	PV1	PV0

If the PWE bit for a particular PWM block is 1, the block is active and controls its assigned port pin; if PWE is 0 the corresponding port pin is controlled by the port. The "value" field (PV5 ... PV0) of each PWM register is compared to (the LS 6 bits of) the common counter. When the value matches, the output FF is cleared, so that the output pin is driven low. When the value rolls over to zero, the output FF is set, so that the output pin is released. Thus the output waveform has a fixed period of 64 PWM counter clocks; its duty cycle is determined by PWMn.5:0.

Three of the nine total PWMs operate as described above; for three others, both the rising and falling edges of the output are delayed by one PWM clock; for the remaining three, both edges are delayed by two PWM clocks. This feature reduces the radio-frequency emission that would otherwise occur when the counter rolled over to zero and all nine open-drain outputs were released.

14-Bit PWM DAC (TDAC)

This feature was partially described in the preceding section. As shown in Figure 3, the 6-bit counter used for the lower precision PWMs is in fact the least significant part of a 14-bit counter used for this facility. The nature of the counter is such that it can achieve a stable output value through its MSB, and the value can propagate through logic like that shown in Figure 3, and the logic output can be stable within one period of the PWM counter clock (e.g., 250 ns) if edge-triggered logic is used to capture the logic output, or within one phase of the PWM counter clock (e.g., 125 ns) if a phase of the PWM counter clock is used to capture the logic output. For cost and die-size reasons, it is preferable that the TDAC counter be a ripple counter.

This feature is controlled by two special function registers:

TDACL

7	6	5	4	3	2	1	0
TD7	TD0	TD1	TD2	TD3	TD4	TD5	TD6

TDACH

7	6	5	4	3	2	1	0
TDE	–	TD13	TD12	TD11	TD10	TD9	TD8

When software wishes to change the 14-bit value (TD0 – TD13), it should first write TDACL and then write TDACH. Alternatively, if the required precision of the duty cycle is satisfied by 6 bits or less, software can simply write TDACH. Note from Figure 3 that this block includes an "extra" 14-bit latch between TDACL/H and the comparator and other logic. The programmed value is clocked into the operative latch when the 7 low-order bits of the counter roll over to zero, provided that the software is not in the midst of loading a new 14-bit value (that is, it is not between writing TDACL and writing TDACH).

In a similar fashion to the lower-precision PWMs, this facility has an output FF that is set when the lower 7 bits of the counter overflow/wrap. The *more* significant 7 bits of the operative latch's programmed value are compared for equality against the *less* significant 7 bits of the counter, and the output FF is cleared when they match. Thus this output has a fixed period of 128 PWM counter clocks, and the duty cycle is determined by the programmed value.

For the higher-precision aspect of this feature, the 7 more-significant bits of the counter are used in a logic block with the 7 less-significant bits of the programmed value. The 7th LSB (binary value 64) of the programmed value is ANDed with the 7th MSB (128) of the counter, the 6th LSB of the value is ANDed with the counter's 6th and 7th MSBs being 10, and so on through the LSB

of the programmed value being ANDed with the counter's 7MSBs being 100000. Then these 7 ANDed terms are ORed. If the result is true/1 at the time the 7 LSBs of the counter match the MSBs of the programmed value, the output is forced high for 1 (additional) PWM counter clock.

The result is that, if the value-64 bit of the 14-bit value is programmed to 1, every other cycle of 128 PWM counter clocks has its duty cycle stretched by one counter clock; if the value-32 bit is programmed to 1, every 4th cycle is stretched, and so on through, if the value-1 bit is programmed to 1, one cycle out of each 128 is stretched.

Assuming the external integrator can handle all this, the net effect is a PWM DAC that has the period of a 7-bit design (which makes the integrator easier and more feasible to design) with the accuracy of a 14-bit one. There is some question whether all of the least significant bits can be effectively integrated, or whether they simply act as a source of ripple in the integrated voltage. An obvious prerequisite for such precision is that the load on the voltage must be very light, like a single op amp or comparator.

The TDAC feature differs from the corresponding features of predecessor parts in several ways:

1. The 14-bit value is functionally composed of major and minor portions of 7 bits each.
2. The 14-bit value is programmed as a contiguous multi-register value that can be manipulated straight-forwardly via arithmetic instructions.
3. As discussed for the 6-bit DACs, both of the preceding parts had a feature whereby the PWM output could be inverted, redundantly with complementing the 14-bit value. This feature has been eliminated.

	ADDRESS TYPE			
	DIRECT	BIT	REGISTER	USE
DATA MEMORY	00-07		R0-R7	On-chip RAM (R0-7 if PSW.4-3 = 00)
	08-0F		R0-R7	On-chip RAM (R0-7 if PSW.4-3 = 01)
	10-17		R0-R7	On-chip RAM (R0-7 if PSW.4-3 = 10)
	18-1F		R0-R7	On-chip RAM (R0-7 if PSW.4-3 = 11)
	20	07-00		On-chip RAM
	21-2E	77-08		On-chip RAM
	2F	7F-78		On-chip RAM
	30-7F			On-chip RAM
SPECIAL FUNCTION REGISTERS	80	87-80	P0	Port 0
	81		SP	Stack Pointer
	82		DPL	Data Pointer LSBYTE
	83		DPH	Data Pointer MSBYTE
	87		PCON	Power Control
	88	8F-88	TCON	Timer Control
	89		TMOD	Timer Mode
	8A		TL0	Timer 0 LSBYTE
	8B		TL1	Timer 1 LSBYTE
	8C		TH0	Timer 0 MSBYTE
	8D		TH1	Timer 1 MSBYTE
	90	97-90	P1	Port 1
	98	9F-98	OSAT	On Screen Attributes
	99		OSDT	On Screen Data
	9A		OSAD	On Screen Address
	A0	A7-A0	P2	Port 2
	A8	AF-A8	IE	Interrupt Enable
	B0	B7-B0	P3	Port 3
	C0	C7-C0	OSCON	On Screen Display Control
	C1		OSMOD	On Screen Display Mode
	C2		OSORG	On Screen Display Origin
	C3		RAMCHR	For Test Use Only
	C4		RAMATT	For Test Use Only
	D0	D7-D0	PSW	Program Status Word
	D2		TDACL	Hi-Res Pulse Width Modulator
	D3		TDACH	Hi-Res Pulse Width Modulator
	D4-D7		PWM0-3	Lo-Res Pulse Width Modulators
	D8	DF-D8	SAD	D/A and Voltage Comparator
	DC-DF		PWM4-7	Lo-Res Pulse Width Modulators
	E0	E7-E0	A	Accumulator
	F0	F7-F0	B	B Register

ON-CHIP RAM IF ACCESSED INDIRECTLY

Figure 1. Data Memory and Special Function Registers on the MTV

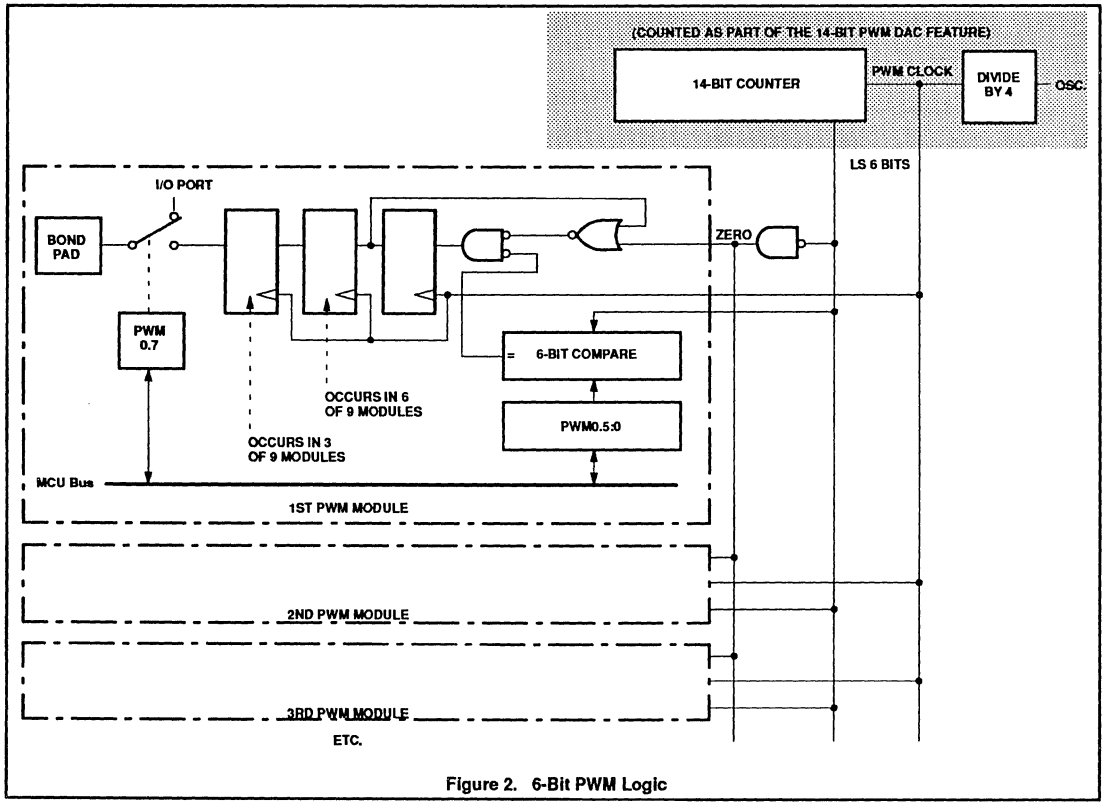


Figure 2. 6-Bit PWM Logic

Software A/D Facility

This facility is shown in Figure 4. It represents an alternate use whereby any of the P1.0 through P1.2 pins can be selected as one input of a linear voltage comparator. The block includes one special function register:

SAD

7	6	5	4	3	2	1	0
VHi	CH1	CH0	St	SAD3	SAD2	SAD1	SAD0

As shown in Figure 4 the other input of the comparator is connected to a 4-bit D/A that is controlled by the 4 LSBs of the SAD register, producing a reference voltage nominally 0.15625V to 4.84375V by steps of 0.3125V. The output of the comparator (high/low) can be read by the program as the MSB of the register, which is bit addressable.

The St bit should be written as 1 in order to initiate a voltage comparison. After writing St=1, the program should include intervening instructions totalling at least six machine cycles (72 CLK periods or 6 microseconds at 12MHz), before the instruction that accesses and tests VHi.

The chan field controls which pin, if any, is connected to this facility:

CH1	CH0	pin
0	0	none
0	1	P1.0
1	0	P1.1
1	1	P1.2

Port 1 has open-drain drivers which will not materially affect an analog voltage as long as any and all pins used for software A/D measurement have corresponding ones in the port register.

On Screen Display (OSD) Module

This block is the largest of the additions that are specific to this product. Its basic function is to superimpose text on the television video image, to indicate various parameters and settings of the receiver or tuner. External circuitry handles the mixing (multiplexing) of the text and the TV video.

The overall OSD block has four input pins: two for a video clock, plus the horizontal and vertical sync signals. The video clock pins are used to connect an LC circuit to an on-chip video oscillator that is independent of the normal MCU clock. The L and C values are chosen so that a video pulse, of a duration equal to the VCLK period, will produce a more-or-less square dot on the screen, that is, a dot having a width approximately equal to the vertical distance between consecutive scan lines.

The video oscillator is stopped (with VCLK2 low) while horizontal sync is asserted, and is released to operate at the trailing edge of horizontal sync. This technique helps provide uniform horizontal positioning of characters/dots from one scan line to the next.

The block has four outputs, three color video signals, and a control signal. Since this block is the major feature of the part, its main inputs and outputs are dedicated pins, without alternate port bits.

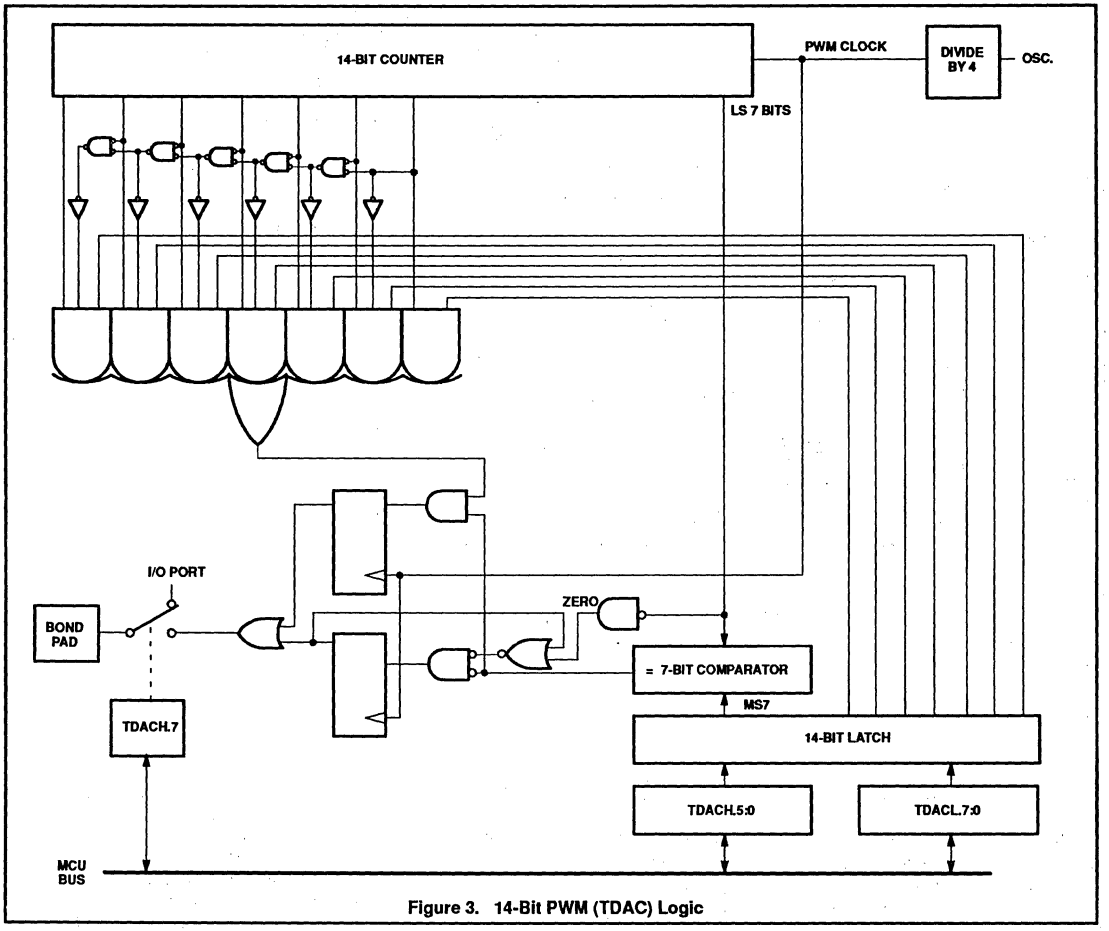


Figure 3. 14-Bit PWM (TDAC) Logic

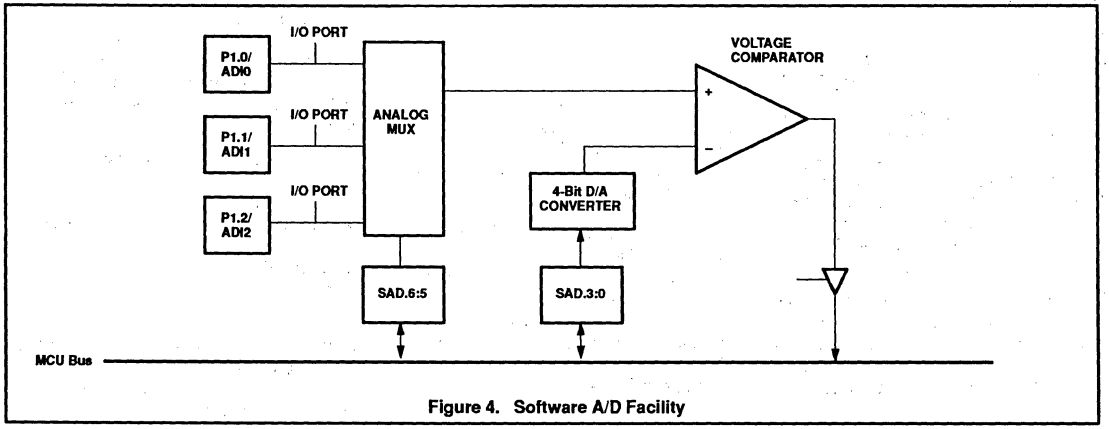


Figure 4. Software A/D Facility

Display RAM

The OSD of the MTV differs from that in preceding devices in one major way: It does not fix the number and size of displayed rows of text. Several predecessor parts allowed two displayed rows of 16 characters each. The MTV simply has 128 locations of display RAM, each of which can contain a displayed character or a New Line character that indicates the end of a row. A variant of the New Line character is used to indicate the end of displayed data.

The three major elements of the OSD facility are shown in the Block Diagram. Each display RAM location includes 6 data bits and 4 attribute bits. The 6 data bits from display RAM, along with a line-within-row count, act as addresses into the character generator ROM, which contains 60 displayable bit maps (64 minus one for each of New Line and three Space characters). Each bit map includes 18 scan lines by 14 dots. The character generator ROM is maskable or programmable along with the program ROM to allow for various character sets and languages.

The programming interface to display RAM is provided by three special function registers:

OSAD

7	6	5	4	3	2	1	0
–	OSAD6	OSAD5	OSAD4	OSAD3	OSAD2	OSAD1	OSAD0

OSDT

7	6	5	4	3	2	1	0
–	–	OSDT5	OSDT4	OSDT3	OSDT2	OSDT1	OSDT0

OSAT (with OSDT = New Line)

7	6	5	4	3	2	1	0
–	–	–	E	–	SR	D	Sh

OSAT (with OSDT = BSpace or SplitBSpace)

7	6	5	4	3	2	1	0
–	–	–	B	–	BC2	BC1	BC0

OSAT (with OSDT = any other)

7	6	5	4	3	2	1	0
–	–	–	B	–	FC2	FC1	FC0

OSAD ("On Screen ADress") contains the address at which data will next be written into display RAM, while the ten active bits in OSDT ("On Screen DaTa") plus OSAT ("On Screen ATtributes") correspond exactly to the 10 bits in each display RAM location. FColor indicates the color of foreground (1) pixels in the ROM bit map for this character, while B indicates whether background (0) pixels should show the current background color (B=1) or television video (B=0). Thus, for the 1 bits in a character's bit map, the VID2:0 pins are driven with (FColor) and VCTRL is driven active, while for 0 bits VID2:0 are driven with the background color (except for shadow bits) and VCTRL is driven with the B bit.

Writing OSAT simply latches the attribute bits into a register, while writing OSDT causes the data bus information, plus the contents of the OSAT register, to be written into display RAM. Thus, for a given display RAM location, OSAT should be written before OSDT. If successive characters are to be written into display RAM with the same attributes, OSAT need not be rewritten for each character, only prior to writing OSDT for the first character with those particular attributes.

In reality, there is a potential conflict between the timing of a write to OSDT and an access to display RAM by the OSD logic for data display. This is resolved by the use of a true dual-ported RAM for display memory.

OSAD is automatically incremented by one after each time OSDT and display RAM are written. Except in special test modes that are beyond the scope of this spec release, display RAM cannot be read by the MCU program.

The OSAT attribute bits associated with the BSpace (data=111110), SplitBSpace (111111), and New Line (111101) characters are interpreted differently from those that accompany other data characters. With BSpace and SplitBSpace, B is interpreted as described above, but the 3 color bits specify the Background color (BColor) for subsequent characters. For BSpace, a change in B and BColor becomes effective at the left edge of the character's bit map. For SplitBSpace, a change in B and BColor occurs halfway through the character horizontally. The normal Space character (111100) has no effect on the BColor value.

BColor values 000 and 111 minimize the occurrence of transient states among the VID2:0 outputs.

The background color defined by the most recently encountered BSpace or SplitBSpace character is maintained on the VID2:0 pins except at the following times:

1. During the active time of HSYNC,
2. During the active time of VSYNC,
3. During those pixels of an active character that correspond to a 1 in the character's bit map,
4. During a "shadow" bit.

The BColor value is not cleared between vertical scans, so that if a single background color is all that is needed in an application, it can be set via a single BSpace character during program initialization, and never changed thereafter. In order for such a BSpace to actually affect the MTV's internal BColor register the Mode field of the OSMOD register must be set to 01 (or higher) so that the OSD hardware is operating.

With a New Line character, if the E bit is 1, no further rows are displayed on the screen. If E is 0 and D is 1, all of the characters in the following row are displayed with Double height and width. If E is 0 and Sh is 1, all of the characters in the following row are displayed with shadowing, as described in a later section. If E is 0 and SR is 1, the next row is a "short row": It is only 4 or 8 scan lines high rather than 18 or 36. Short rows can be used for underlined text.

The latches in which the E, D, Sh, and SR bits are captured are cleared to zero at the start of each vertical scan. This means that if the first text line on the screen is a short row, or if it contains either double size or shadowing, the text must be preceded by a New Line character. Like all such characters, this initial New Line advances the vertical screen position; the VStart value (see below) should take this fact into account.

Other OSD Registers

A number of changes in the OSD architecture have reduced the number of other special function registers involved in the feature, below the number needed with predecessor devices:

1. The elimination of certain options such as 4, 6, or 8X character sizes and alternate use of two of the video outputs.
2. The moving of certain other options from central registers to display RAM, such as foreground color codes and background selection.

OSCON

7	6	5	4	3	2	1	0
IV	Pv	Lv	Ph	Pc	Po	DH	BFe

The IV bit is the interrupt flag for the OSD feature. It is set by the leading edge of the VSYNC pulse, and is cleared by the hardware when the VSYNC interrupt routine is vectored to. It can also be set or cleared by software writing a 1 or 0 to this bit.

NOTE

It is theoretically possible that a VSYNC interrupt could be missed, or an extra one generated, if OSCON is read, then modified internally (e.g., in Ac), and the result written back to OSCON. However, none of the other bits in OSCON are reasonable candidates for dynamic change. Special provisions are included in the MTV logic so that IV will not be changed by a single "read-modify-write" instruction such as SETB or CLR, unless the instruction specifically changes IV.

A 0 (1) in Pv designates that the VSYNC input is high-active (low-active). One effect of this bit is that the VID2:0 and VCTRL outputs are blocked (held at black/inactive) during the active time of VSYNC. The IV bit is set on the leading edge of the VSYNC pulse; thus Pv controls whether the OSD interrupt occurs in response to a high-to-low or low-to-high transition on VSYNC.

A 0 (1) in Lv designates that the leading edge (active level) of VSYNC, as defined by Pv, clears the state counter that is used to determine the vertical start of on-screen data. In effect, Lv=0(1) says that the leading (trailing) edge of VSYNC is the time reference for the video field.

A 0 (1) in Ph designates that the HSYNC input is high-active (low-active).

A 0 (1) in Pc designates that a high (low) on the VCTRL output means "show the color on VID2:0".

A 0 (1) in Po designates that a 0 (1) internal to the MTV corresponds to a low on one of the VID2:0 pins. This control bit is needed only because the Shadowing feature needs to generate black pixels without reference to a register value: Internally, the 3-bit code 000 always designates black.

If DH is 1, character sizes are doubled vertically but not horizontally. This feature allows the MTV to be used in "improved definition" systems that are not interlaced. The vertical doubling imposed by DH does not affect the VStart logic described below: It operates in HSync units regardless of DH or D.

If BFe is 1, the BF output tracks whether each bit in displayed characters is a foreground bit (low) or a background bit (high). If BFe is 0, the BF pin remains high.

OSORG

7	6	5	4	3	2	1	0
HS4	HS3	HS2	HS1	HS0	VS2	VS1	VS0

The HStart field (HS4 – HS0) defines the left end (start) of all of the on-screen character rows, as a multiple of four VCLKs. Active display begins 4(HStart)+1 VCLKs plus one single-sized character width after the trailing edge of HSYNC. Counting variations in Wc, there may be 17 to 143 VCLKs from the end of HSYNC to the start of the first character of each row.

The VStart field (VS2 – VS0) defines the top (start) of the first on-screen character row, as a multiple of four HSYNC pulses. Active display begins 4(VStart)-1 HSYNCs after the field's time reference point, a range of 3 to 31. Subsequent character rows occur directly below the first, such that the last scan line of one row is directly followed by the first scan line of the next row. Successive New Line characters (with or without the Short Row designation) can be used to vertically separate text rows on the screen.

Neither the HStart nor VStart parameter is affected by the D line attribute that is used to display double-sized characters.

OSMOD

7	6	5	4	3	2	1	0
Wc	–	Mode1	Mode0	–	SHM2	SHM1	SHM0

If the mode bits (Mode 1, Mode 0) are 00, the OSD feature is disabled. The VCLK oscillator is disabled, VID2:0 are set to black, and VCTRL is held inactive. This is the mode to which the MTV OSD logic is reset. A direct transition from this mode to active display (1x) would result in undefined operation and visual effects for the duration of the current video field (until the next VSYNC).

If the mode is 01, the VCLK oscillator is enabled and the OSD logic operates normally internally, but VID2:0 are set to black and VCTRL is held inactive. The OSD feature can be toggled between this state and 1x as desired to achieve real-time special effects such as "vertical wiping."

Mode 10 represents normal OSD operation. Active characters can be shown against TV video (for characters with B=0) or (for characters with B=1) against a background of the color defined as an attribute of BSpace and SplitBSpace characters.

In mode 11, characters can be displayed but all of the receiver's normal video is inhibited by holding VCTRL asserted throughout the active portion of each scan line. Since VID2:0 are driven with the current background color during this time, except during the foreground portion of displayed characters, this produces text against a solid background. This mode is useful for extensive displays that require user concentration.

If Wc is 1, then each displayed character is horizontally terminated after 12 bits have been output, as opposed to after 14 bits if Wc is 0. This allows text to be "packed" more tightly so that more characters can be displayed per line. In effect, the 2 bits out of the display ROM, which would otherwise be the rightmost 2 of the 14, are ignored when Wc is 1. Clearly, if this feature is to be used, it must be accounted for in the design of the bit maps in the display ROM.

The 3-bit ShMode field (SHM2 – SHM0) determines how characters are shadowed in rows for which the SH row attribute is 1. As shown in Figure 5, the values 000-110 indicate an apparent light source position ranging from the lower left clockwise to the lower right, while the value 111 indicates full-surround shadowing.

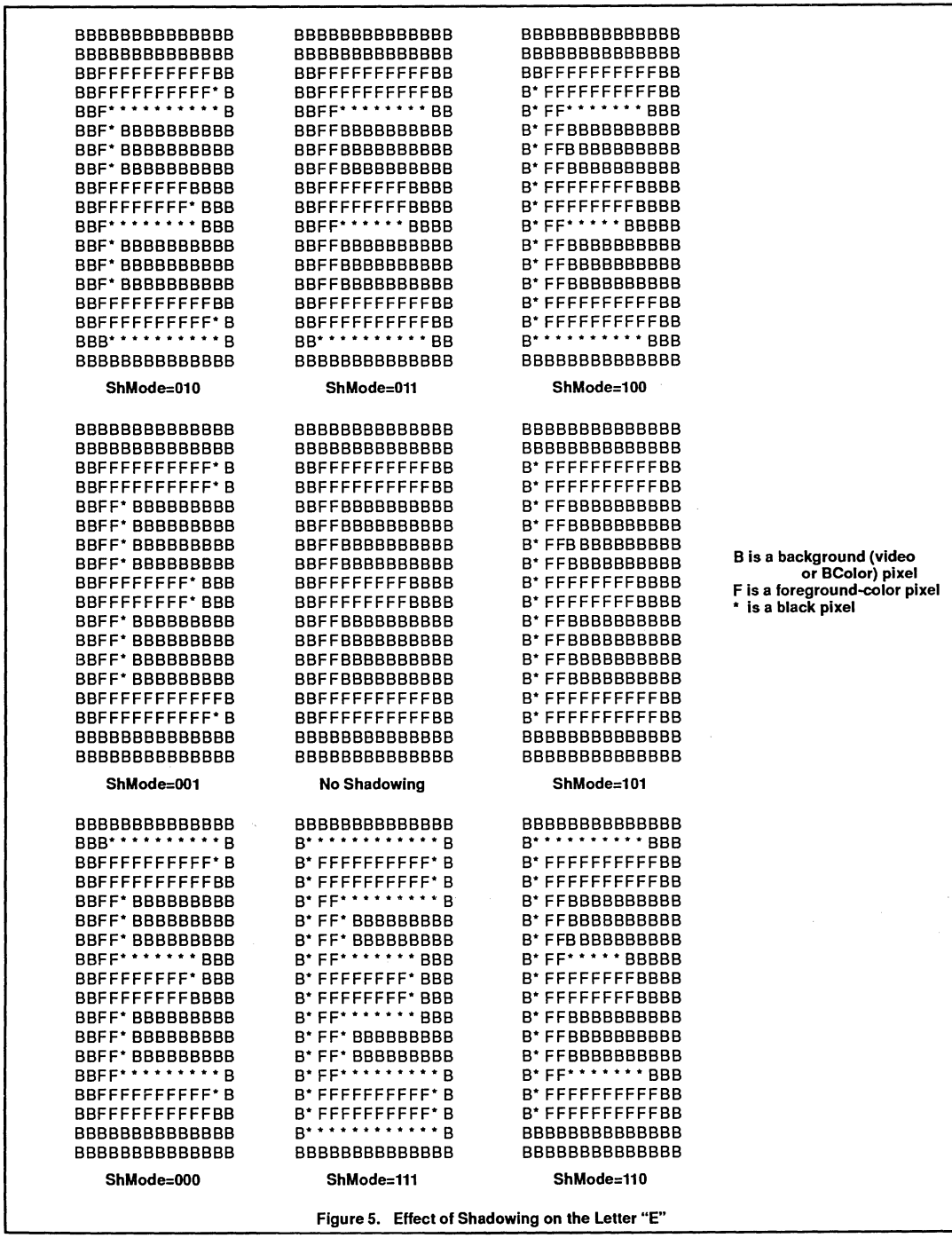


Figure 5. Effect of Shadowing on the Letter "E"

DC ELECTRICAL CHARACTERISTICS

V_{CC} = 5V ±10%, T_{amb} = 0°C to +70°C

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNIT	NOTES
			MIN	MAX		
V_{IL}	Input low voltage		–0.5	$0.2V_{CC}$–0.1	V	
V_{IL1}	Input low voltage (VSYNC)	Neg. HSYNC polarity (OSPH = 1)	–0.5	$0.16 \times V_{CC}$	V	5
		Pos. HSYNC polarity (OSPH = 0)	–0.5	$0.20 \times V_{CC}$	V	5
V_{IH1}	Input high voltage (P1.2:0, P2.7:0, P3.6:5, P3.3:1, VSYNC, HSYNC)		$0.2V_{CC}$+0.9	V_{CC}+0.5	V	
V_{IH2}	Input high voltage (port 0, P1.3, P3.7, P3.4, P3.0)	I_{IH} = 2mA	$0.2V_{CC}$+0.9	12.6	V	
V_{IH3}	Input high voltage (VSYNC)		$0.6 \times V_{CC}$	V_{CC} + 0.5V	V	
V_{IH} – V_{CC}	Input high voltage (port 0, P1.3, P3.7, P3.4, P3.0) with respect to V_{CC}			8	V	1
I_{IN}	Input current (VSYNC)	Neg. HSYNC polarity (OSPH = 1)	–80	+120	µA	5
		Pos. HSYNC polarity (OSPH = 0)	0	+10	µA	5
V_{IH}	Input high voltage (XTAL1, VCLK1, RST)		$0.7V_{CC}$	V_{CC}+0.5	V	
V_{OL1}	Output low voltage (P2.3:0, P3.6:5)	I_{OL} = 10mA		0.5	V	
V_{OL2}	Output low voltage (TDAC, PWM0:7)	I_{OL} = 700µA		0.5	V	2
V_{OL3}	Output low voltage (all other outputs)	I_{OL} = 1.6mA		0.45	V	
V_{OH}	Output high voltage (port 1, VID2:0, VCTRL, BF)	I_{OH} = –60µA	2.4		V	
R_{RST}	Reset pulldown resistor		50	300	kΩ	
C_{IO}	Pin capacitance	Test freq = 1MHz, T_{amb} = 25°C		10	pF	
I_{PD}	Power-down current	V_{CC} = 2 to 6V		5	mA	
I_{CC}	Normal mode supply current	V_{CC} = 5.5V		30	mA	3
HYS	Hysteresis (VSYNC)	Either HSYNC polarity	0.445		V	

NOTES:

1. This maximum applies at all times, including during power switching, and must be accounted for in power supply design. During a power-on process, the +12 volt source used for external pullup resistors should not precede the V_{CC} of the MTV up their respective voltage ramps by more than this margin, nor, during a power-down process, should V_{CC} precede +12V down their respective voltage ramps by more than this margin.
2. The specified current rating applies when any of these pins is used as a Pulse Width modulated output. For use as a port output, the rating is as given subsequently.
3. I_{CC} measured with OSD block initialized and Reset remaining low.
4. The capacitance of pins P0.0 and P0.7 for the 87C054 exceeds 10pF. P0.0 is 40pF maximum, while P0.7 is 20pF maximum.
5. Input current for negative HSYNC polarity is a switching current. This current may alternate between positive current (into the pin) and negative (out of the pin) during the same HSYNC transition.

AC ELECTRICAL CHARACTERISTICS

V_{CC} = 5V ±10%, T_{amb} = 0°C to +70°C

SYMBOL	PARAMETER	TENTATIVE LIMITS		UNIT	NOTES
		MIN	MAX		
$1/t_{CLCL}$	XTAL Frequency	6	12	MHz	1
t_{CHCX}	XTAL1 Clock high time	20		ns	2
t_{CLCX}	XTAL1 Clock low time	20		ns	2
t_{CLCH}	XTAL1 Clock rise time		20	ns	2
t_{CLCL}	XTAL1 Clock fall time	5	20	ns	2
$1/t_{VCLCL}$	VCLK Frequency	5	8	MHz	
$\lvert t_{VCOH}-t_{VCOL} \rvert$	Rise vs. fall time skew on any one of VID2:0, VCTRL, BF		40	ns	3
$\lvert t_{VCOH1}-t_{VCOH2} \rvert$	Rise time skew between any two of VID2:0, VCTRL, BF		30	ns	3
$\lvert t_{VCOL1}-t_{VCOL2} \rvert$	Fall time skew between any two of VID2:0, VCTRL, BF		30	ns	3

NOTES:
1. The MTV is tested at its maximum XTAL frequency, but not at any other (lower) rate.
2. These parameters apply only when an external clock signal is used.
3. These parameters assume equal loading at C_L = 100pF, for all the referenced outputs. These parameters are specified but not tested.

PROGRAMMING CONSIDERATIONS

EPROM Characteristics

The 87C054 is programmed by using a modified Quick-Pulse Programming algorithm similar to that used for devices such as the 87C51. It differs from these devices in that a serial data stream is used to place the 87C751 in the programming mode.

Figure 6 shows a block diagram of the programming configuration for the 87C054. Port pin P0.0 is used as the programming voltage supply input (V_{PP} signal). Port pin P0.1 is used as the program (PGM/) signal. This pin is used for the 25 programming pulses.

Port 2 is used as the address input for the byte to be programmed and accepts both the high and low components of the eleven bit address. Multiplexing of these address components is performed using the ASEL input. The user should drive the ASEL input high and then drive port 2 with the high order bits of the address. ASEL should remain high for at least 13 clock cycles. ASEL may then be driven low which latches the high order bits of the address internally. The high address should remain on port 2 for at least two clock cycles after ASEL is driven low. Port 2 may then be driven with the low byte of the address. The low address will be internally stable 13 clock cycles later. The address will remain stable provided that the low byte placed on port 2 is held stable and ASEL is kept low. **Note:** ASEL needs to be pulsed high only to change the high byte of the address.

Port 3 is used as a bidirectional data bus during programming and verify operations. During programming mode, it accepts the byte to be programmed. During verify mode, it provides the contents of the EPROM location specified by the address which has been supplied to Port 2.

The XTAL1 pin is the oscillator input and receives the master system clock. This clock should be between 1.2 and 6MHz.

The RESET pin is used to accept the serial data stream that places the 87C054 into various programming modes. This pattern consists of a 10-bit code with the LSB sent first. Each bit is synchronized to the clock input, X1.

Programming Operation

Figures 7 and 8 show the timing diagrams for the program/verify cycle. RESET should initially be held high for at least two machine cycles. P0.1 (PGM/) and P0.0 (V_{PP}) will be at V_{OH} as a result of the RESET operation. At this point, these pins function as normal quasi-bidirectional I/O ports and the programming equipment may pull these lines low. However, prior to sending the 10-bit code on the RESET pin, the programming equipment should drive these pins high (V_{IH}). The RESET pin may now be used as the serial data input for the data stream which places the 87C054 in the programming mode. Data bits are sampled during the clock high time and thus should only change during the time that the clock is low. Following transmission of the last data bit, the RESET pin should be held low.

Next the address information for the location to be programmed is placed on port 2 and ASEL is used to perform the address multiplexing, as previously described. At this time, port 1 functions as an output.

A high voltage V_{PP} level is then applied to the V_{PP} input (P0.0). (This sets Port 1 as an input port). The data to be programmed into the EPROM array is then placed on Port 3. This is followed by a series of programming pulses applied to the PGM/ pin (P0.1). These pulses are created by driving P0.1 low and then high. This pulse is repeated until a total of 25 programming pulses have occurred. At the conclusion of the last pulse, the PGM/ signal should remain high.

The V_{PP} signal may now be driven to the V_{OH} level, placing the 87C054 in the verify mode. (Port 3 is now used as an output port). After four machine cycles (48 clock periods), the contents of the addressed location in the EPROM array will appear on Port 3.

The next programming cycle may now be initiated by placing the address information at the inputs of the multiplexed buffers, driving the V_{PP} pin to the V_{PP} voltage level, providing the byte to be programmed to Port 3 and issuing the 26 programming pulses on the PGM/ pin, bringing V_{PP} back down to the V_{OH} level and verifying the byte. (See Table 3.)

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Flourless part number 2345–5 or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-s/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000µW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 3. Implementing Program/Verify Modes

OPERATION	SERIAL CODE	P0.1 (PGM/)	P0.0 (V_{PP})
Program user EPROM	286H	–*	V_{PP}
Verify user EPROM	286H	V_{IH}	V_{IH}

NOTE:
* Pulsed from V_{IH} to V_{IL} and returned to V_{IH}.

EPROM PROGRAMMING AND VERIFICATION

T_{amb} = 21°C to +27°C, V_{CC} = 5V ±10%, V_{SS} = 0V

SYMBOL	PARAMETER	MIN	MAX	UNIT
$1/t_{CLCL}$	Oscillator/clock frequency	1.2	6	MHz
t_{AVGL}*	Address setup to P0.1 (PROG–) low	10µs + 24t_{CLCL}		
t_{GHAX}	Address hold after P0.1 (PROG–) high	48t_{CLCL}		
t_{DVGL}	Data setup to P0.1 (PROG–) low	38t_{CLCL}		
t_{DVGL}	Data setup to P0.1 (PROG–) low	38t_{CLCL}		
t_{GHDX}	Data hold after P0.1 (PROG–) high	36t_{CLCL}		
t_{SHGL}	V_{PP} setup to P0.1 (PROG–) low	10		µs
t_{GHSL}	V_{PP} hold after P0.1 (PROG–)	10		µs
t_{GLGH}	P0.1 (PROG–) width	90	110	µs
t_{AVQV}**	V_{PP} low (V_{CC}) to data valid		48t_{CLCL}	
t_{GHGL}	P0.1 (PROG–) high to P0.1 (PROG–) low	10		µs
t_{SYNL}	P0.0 (sync pulse) low	4t_{CLCL}		
t_{SYNH}	P0.0 (sync pulse) high	8t_{CLCL}		
t_{MASEL}	ASEL high time	13t_{CLCL}		
t_{MAHLD}	Address hold time	2t_{CLCL}		
t_{HASET}	Address setup to ASEL	13t_{CLCL}		
t_{ADSTA}	Low address to address stable	13t_{CLCL}		

NOTES:
* Address should be valid at least 24t_{CLCL} before the rising edge of P0.0 (V_{PP}).
** For a pure verify mode, i.e., no program mode in between, t_{AVQV} is 14t_{CLCL} maximum.

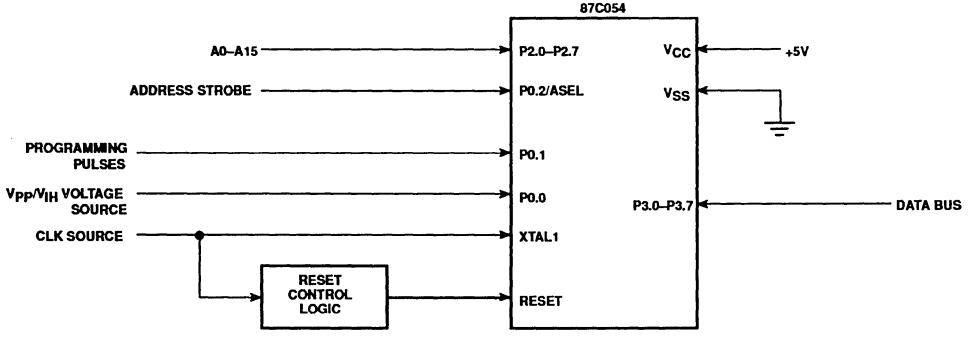


Figure 6. Programming Configuration

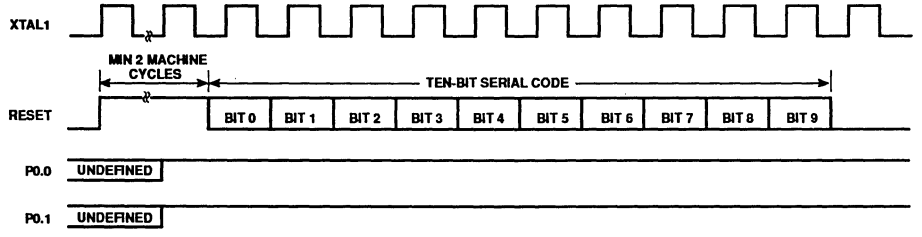


Figure 7. Entry into Program/Verify Modes

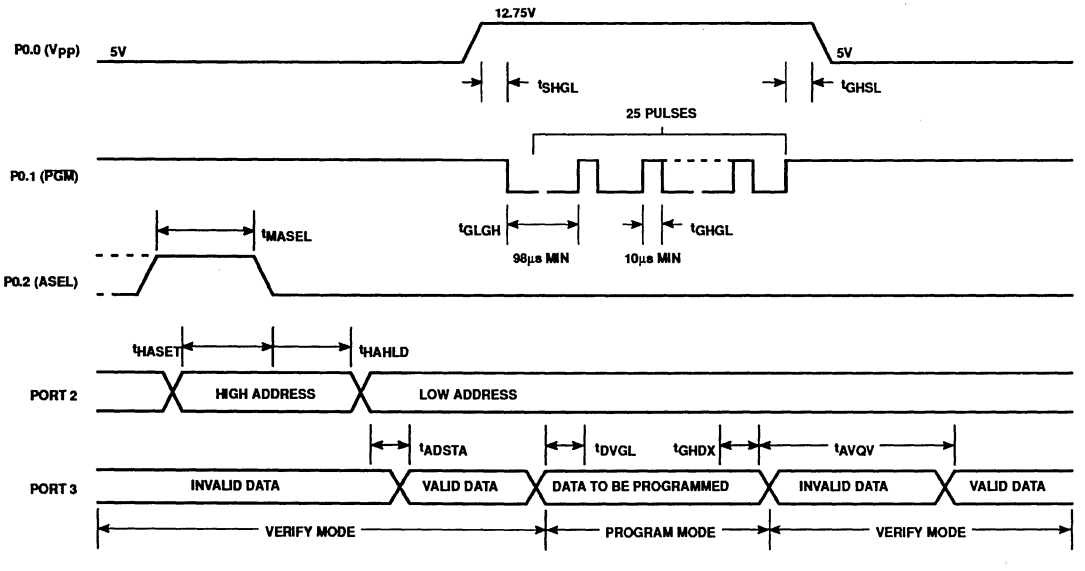


Figure 8. Program/Verify Cycle

8XCL410 OVERVIEW

The 80CL410 (ROMless version) and 83CL410 (ROM version) are referred to collectively in this overview as the 8XCL410. The 8XCL410 is socket compatible with the 80C51 and has many of the same features, but at a much lower power and clock frequency.

The 8XCL410 is the first member of a family of low-power 80C51 derivative parts. The features of the 8XCL410 include:

- 4k × 8 ROM
- 128 × 8 RAM
- Two 16-bit timer/counters
- A two level nested priority interrupt structure
- An I²C serial interface
- Thirteen interrupt sources (with eight additional external interrupts)
- Idle and power-down modes
- Interrupt or reset return from power-down
- Six oscillator configurations
  - Quartz crystal
  - Ceramic resonator
  - RC
  - LC
  - External
- Input supply range from 1.8V to 6V
- Operating frequency from DC to 16MHz

The 8XCL410 can be operated from a single battery supply which can vary between 1.8V and 6V, and for an AC supply the part requires only a simple voltage regulator. In some cases the part can be operated from an unstabilized supply, eliminating altogether the need for a regulator.

The power consumption of the part is much lower than that of a standard 80C51. Operating at 3.5MHz from a 3V supply, the 8XCL410 typically draws less than 1mA of current. The power consumption of the part is directly related to the supply voltage and clock frequency. As the supply voltage or clock frequency are increased, the power consumption also increases. At 12MHz and a supply voltage of 5V, the 8XCL410 will draw about 10mA of current, which is slightly less than the current that an 80C51 would require.

The advantage that the 8XCL410 has over the 80C51 is in its ability to operate at very low frequencies and supply voltages. This makes the part an ideal choice for applications where power is supplied from batteries, or where low supply voltages or clock frequency are necessary. The 8XCL410 features a fully static design. Using the on-chip oscillator, the clock frequency is limited to a minimum of 32kHz, but using an external oscillator the part can be operated down to DC. This means that the clock can be turned off, and when it is started again the microcontroller will continue with the action that it was performing when the clock was stopped. This is something that is impossible with dynamic devices because their internal nodes must be constantly refreshed. The static design of the 8XCL410 offers the user the ultimate in power-down modes, because the part can be stopped until it is needed and then started from where it was at when it was stopped, with no loss of internal states or data. The power consumption of the 8XCL410 when the clock is stopped is less than 1μA.

Differences from the 80C51

Special Function Registers

The 8XCL410 contains most of the special function registers found in the 80C51 as well as eight additional SFRs that have been added to handle the I²C serial interface and eight additional external interrupts. The standard UART found on the 80C51 has been replaced with an I²C serial interface, so the SFRs SCON and SBUF have been removed. Four SFRs have been added to handle the I²C interface; they are: S1CON, S1DAT, S1STA, and S1ADR.

The interrupt structure on the 8XCL410 has been upgraded to include eight additional external interrupts. The IE and IP registers on the 80C51 have had their names changed on the 8XCL410 to IEN0 and IP0. In addition, two SFRs have been added to handle the additional external interrupts. The registers are IEN1 and IP1.

Two more SFRs are added to allow the user to set the polarity of the additional external interrupts and to hold the interrupt request flags for those added interrupts. The SFRs are the interrupt polarity register (IX1) and the interrupt request flag register (IRQ1).

Table 1 shows the special function registers, their locations, and their reset values for the 8XCL410.

I²C Serial Interface

The serial port supports the two-wire I²C bus. The I²C bus consists of a data line (SDA) and a clock line (SCL). These lines are multiplexed functions of I/O port pins P1.7 and P1.6, respectively. The main features of the bus are:

- Bidirectional data transfer between masters and slaves
- True multimaster bus
- Arbitration between simultaneously transmitting masters without loss or corruption of the serial data on the bus
- Synchronized clock allows devices with different bit rates to communicate
- The serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer.

The CPU of the 8XCL410 interfaces to the I²C logic via four special function registers. The registers are S1CON (I²C control register), S1DAT (data register), S1STA (status register), and S1ADR (slave address register).

A detailed discussion of the I²C bus is given in section 2 of this users' guide. The I²C interface used on the 8XCL410 is functionally identical to the one on the 8XC552. A detailed discussion of this I²C hardware is given in the 8XC552 section, so the discussion here will be limited to a review of each of the four I²C special function registers. A block diagram of the I²C serial interface is shown in Figure 1.

The functions of the I²C interface are controlled by the S1CON register.

Table 1. 8XCL410 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes):										
DPH	High byte	83H									00H
DPL	Low byte	82H									00H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP0*#	Interrupt priority 0	B8H	–	–	PS1	–	PT1	PX1	PT0	PX0	xx000000B
			FF	FE	FD	FC	FB	FA	F9	F8	
IP1*#	Interrupt priority 1	F8H	PX9	PX8	PX7	PX6	PX5	PX4	PX3	PX2	00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IEN0*#	Interrupt enable 0	A8H	EA	–	ES1	–	ET1	EX1	ET0	EX0	00H
			EF	EE	ED	EC	EB	EA	E9	E8	
IEN1*#	Interrupt enable 1	E8H	EX9	EX8	EX7	EX6	EX5	EX4	EX3	EX2	00H
			C7	C6	C5	C4	C3	C2	C1	C0	
IRQ1*#	Interrupt request flag	C0H	IQ9	IQ8	IQ7	IQ6	IQ5	IQ4	IQ3	IQ2	00H
IX1#	Interrupt polarity	E9H									00H
P0*	Port 0	80H	87	86	85	84	83	82	81	80	FFH
P1*	Port 1	90H	97	96	95	94	93	92	91	90	FFH
P2*	Port 2	A0H	A7	A6	A5	A4	A3	A2	A1	A0	FFH
P3*	Port 3	B0H	B7	B6	B5	B4	B3	B2	B1	B0	FFH
PCON	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
S1ADR#	Slave address	DBH									00H
			DF	DE	DD	DC	DB	DA	D9	D8	
S1CON*#	Serial control	D8H	–	ENS1	STA	STO	SI	AA	CR1	CR0	x0000000B
S1DAT#	Serial data	DAH									00H
S1STA#	Serial status	D9H									11111000B
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer/counter control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
TMOD	Timer/counter mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
TH0	Timer 0 high byte	8CH									00H
TH1	Timer 1 high byte	8DH									00H
TL0	Timer 0 low byte	8AH									00H
TL1	Timer 1 low byte	8BH									00H

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.

S1CON (D8H)

7	6	5	4	3	2	1	0
–	ENS1	STA	STO	SI	AA	CR1	CR0

CR0 and CR1—Clock Rate bits. These bits determine the serial clock (SCL) frequency when the 8XCL410 is in the master mode. The various SCL serial clock rates that can be achieved are shown in Table 2.

The SCL rate generated when both CR0 and CR1 are low is usually used when the I²C interface on other parts are software driven and slow. The maximum SCL rate is 100kHz and can be derived from either a 12MHz or 6MHz oscillator. A variable bit rate can be used if timer 1 is not required for another purpose while the 8XCL410 is in the master mode.

The CR0 and CR1 bits have no effect when the part is in the slave mode, because SCL is generated only by the bus master. In the slave mode, the part will automatically synchronize with the I²C bus clock frequency.

AA—Assert Acknowledge. When the AA bit is set (1), an acknowledge will be returned when the 8XCL410 recognizes its own slave address, recognizes the general call address if S1ADR.0 is set (1), or receives a data byte while it is either the bus master or the selected slave. If AA is not set (0), then no acknowledge will be returned for any condition. The I²C bus hardware is not disabled, but the part will not respond to its own slave address or the general call address (even is S1ADR is set), nor will it acknowledge received bytes.

SI—Serial Interrupt flag SI is set by hardware when one of 25 of the 26 possible I²C hardware states on the 8XCL410 is entered. The only state that does not cause SI to be set is F8H, which indicates that no relevant state information is available. An interrupt will only be requested while SI is set (1) and the serial interrupt is enabled in the IEN0 (interrupt enable) SFR. When SI is set, the low period of the I²C clock (SCL) is stretched (that is, held low) until SI is cleared. SI can only be cleared by software.

STO—STOp flag. When the 8XCL410 is in the master mode and STO is set (1), a stop condition will be forced on the I²C bus by the part. When the stop is detected on the bus by the 8XCL410's hardware, it will clear the STO flag.

STA—STArt flag. When the 8XCL410 is set to enter the master mode and STA is set (1), the part will check the status of the I²C bus and generate a START condition if the bus is free. If the bus is not free, the 8XCL410 will wait for a STOP condition on the I²C bus and then after a half period delay (of SCL) it will generate a START.

If both STA and STO are set, and the part is in the master mode, a STOP will be forced on the bus, and then following that with the appropriate delays, a START will be forced.

ENS1—Enable I²C Serial Port. When ESN1 is low, the part will not respond to its address or any activity on the I²C bus. The SDA and SCL outputs are in a high impedance state. P1.6 and P1.7 can be used as open drain port pins. When ENS1 is set (1), the I²C serial port is enabled. Port latches P1.6 and P1.7 must be set (1).

ENS1 should not be used to temporarily release the bus, because the I²C bus status S1STA is cleared and the part's bus status lost when ESN1 is reset (0). To temporarily idle the bus, the AA flag should be used.

The data to be transmitted to or received from the I²C bus is written into or read from the S1DAT register.

S1DAT (DAH)

7	6	5	4	3	2	1	0
SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0

SD7-SD0—Serial Data bits. A byte to be transmitted is written into this register, and a byte received is read from this register. A 1 in the register corresponds to a high level on the bus, and a 0 corresponds to a low level. Data shifts into or out of the register from left to right.

The status of the bus can be determined at any time by reading the S1STA register. This is a read only register in which the three least significant bits are always zero (0).

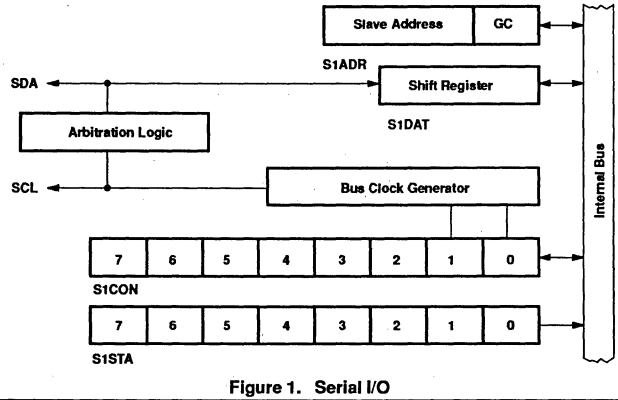


Figure 1. Serial I/O

Table 2. SCL Frequency (kHz) at f_{OSC}

CR1	CR0	6MHz	12MHz	16MHz	f_{OSC} DIVIDED BY
0	0	6.25	12.5	17	960
0	1	50	100	NA	120
1	0	100	NA	NA	60
1	1	0.25 < 31.25	0.5 < 62.5	0.65 < 83.3	96 × (256 – Timer 1 reload range 0–254 in mode 1)

S1STA (D9H)

7	6	5	4	3	2	1	0
SC4	SC3	SC2	SC1	SC0	0	0	0

SC4-SC0—Status Code bits. These bits hold a status code that indicates the current status of the bus. The contents of these bits can be used to vector to a service routine, which optimizes the response time of the software and consequently that of the I^2C bus.

The following is a list of the status codes for each S1STA value.

Master Transmitter Mode

08H – A START condition has been transmitted.
10H – A repeated START condition has been transmitted.
18H – Slave address and write bit transmitted, acknowledge received.
20H – Slave address and write bit transmitted, acknowledge not received.
28H – Data transmitted, acknowledge received.
30H – Data transmitted, acknowledge not received.
38H – Arbitration lost while transmitting slave address, R/W bit, or data.

Master Receiver Mode

38H – Arbitration lost while returning acknowledge.
40H – Slave address and read bit transmitted, acknowledge returned.
48H – Slave address and read bit transmitted, acknowledge not returned.
50H – Data received, acknowledge returned.
58H – Data received, acknowledge not returned.

Slave Receiver Mode

60H – Own slave address and write bit received, acknowledge returned.
68H – Arbitration lost. Own slave address and write bit received, acknowledge returned.
70H – General call received, acknowledge returned.
78H – Arbitration lost. General call received.
80H – Received own slave address and data byte, acknowledge returned.
88H – Received own slave address and data byte, acknowledge not returned.
90H – Received general call and data byte, acknowledge returned.
98H – Received general call and data byte, acknowledge not returned.
A0H – Stop or repeated start received while still addressed as slave transmitter or receiver.

Slave Transmitter Mode

A8H – Own slave address and read bit received, acknowledge returned.
B0H – Arbitration lost. Own slave address and read bit received, acknowledge returned.
B8H – Data byte transmitted, acknowledge received.
C0H – Data byte transmitted, acknowledge not received.
C8H – Last data byte transmitted, acknowledge received.

All Modes

00H – Bus error due to an erroneous start or stop condition

The slave address that the part is to respond to is put into the S1ADR special function register.

S1ADR (DBH)

7	6	5	4	3	2	1	0
SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0

SA7-SA1—Slave Address bits. The 7-bit slave addresses that the part is to respond to is loaded into these seven bits.

SA0—This bit can be set so that the part will respond to a general call address on the I^2C bus. Clearing (0) this bit will prevent the part from responding to a general call address.

The Interrupt Structure

External events and the real-time-driven on-chip peripherals require service by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution, a multiple-source, two-priority level, nested interrupt system is provided. The 8XCL410 acknowledges interrupt requests from 13 sources as follows:

Priority	Source	Vector Address	Function
Highest	INT0	0003H	External interrupt 0
	S1	002BH	I^2C serial interrupt
	INT5	0053H	External interrupt 5
	T0	000BH	Timer 0 interrupt
	INT6	005BH	External interrupt 6
	INT1	0013H	External interrupt 1
	INT2	003BH	External interrupt 2
	INT7	0063H	External interrupt 7
	T1	001BH	Timer 1 interrupt
	INT3	0043H	External interrupt 3
	INT8	006BH	External interrupt 8
	INT4	004BH	External interrupt 4
Lowest	INT9	0073H	External interrupt 9

There are six special function registers associated with the interrupt portion of the 8XCL410. They are IEN0, IP0, IEN1, IP1, IX1, and IRQ1. Following is a detailed description of each.

IEN0—Interrupt Enable register zero. This register has the same function as the IE register found on the 80C51. The only difference between the two is that the bit that controls the serial interface interrupt has been moved from bit 4 to bit 5. This has been done because the 8XCL410 has an I^2C serial interface and bit 5 is used on other parts for the I^2C interrupt enable bit.

IEN0 (A8H)

7	6	5	4	3	2	1	0
EA	–	ES1	–	ET1	EX1	ET0	EX0

EA – General Enable/Disable control
0 = All interrupts disabled.
1 = Interrupts can be individually enabled or disabled.
ES1 – I^2C interrupt enable.
(1 = enabled, 0 = disabled)
ET1 – Timer 1 interrupt enable.
(1 = enabled, 0 = disabled)
EX1 – External interrupt 1 enable.
(1 = enabled, 0 = disabled)
ET0 – Timer 0 interrupt enable.
(1 = enabled, 0 = disabled)
EX0 – External interrupt 0 enable.
(1 = enabled, 0 = disabled)

IP0—Interrupt Priority register zero. This register has the same function as the IP register on the 80C51. The only difference is that the serial interface priority is set on bit 5.

IP0 (B8H)

7	6	5	4	3	2	1	0
–	–	PS1	–	PT1	PX1	PT0	PX0

PS1 – I^2C interrupt priority.
(1 = high, 0 = low)
PT1 – Timer 1 interrupt priority.
(1 = high, 0 = low)
PX1 – External interrupt 1 priority.
(1 = high, 0 = low)
PT0 – Timer 0 interrupt priority.
(1 = high, 0 = low)
PX0 – External interrupt 0 priority.
(1 = high, 0 = low)

IEN1—Interrupt Enable register one. This register contains the interrupt enables for the eight external interrupts that have been added to the 8XCL410. Clearing (0) the bit in the IEN1 register disables all of the interrupts in this register as well as those in the IEN0 register.

IEN1 (E8H)

7	6	5	4	3	2	1	0
EX9	EX8	EX7	EX6	EX5	EX4	EX3	EX2

EX9-EX2—External interrupt enables for external interrupts 2 – 9. (0 = disabled, 1 = enabled)

IP1—Interrupt priority register one. This register allows the priority level of each of the additional external interrupt enables to be set.

IP1 (D8H)

7	6	5	4	3	2	1	0
PX9	PX8	PX7	PX6	PX5	PX4	PX3	PX2

PX9-PX2—External interrupt priority for external interrupts 2 – 9. (0 = low, 1 = high)

IX1—Interrupt Polarity register. This register allows the programmer to determine the polarity of external interrupts 2 – 9 that will be sensed for the interrupt. If the bit for an interrupt is set to 1, then the interrupt will be triggered by a high input on that external interrupt. If the bit is cleared to 0, then the interrupt will be triggered by a low on that external interrupt.

IX1 (E9H)

7	6	5	4	3	2	1	0
IL9	IL8	IL7	IL6	IL5	IL4	IL3	IL2

IL9-IL2—External Interrupt polarity bits corresponding to external interrupts 9 – 2. (1 = high trigger, 0 = low trigger)

IRQ1—Interrupt Request flag register. This register contains flags that are set when one of the external interrupts 2 – 9 are requested. The flags will only be set if the corresponding interrupt is enabled in the IE1 register. The flags must be cleared by software.

IRQ1 (C0H)

7	6	5	4	3	2	1	0
IQ9	IQ8	IQ7	IQ6	IQ5	IQ4	IQ3	IQ2

IQ9-IQ2—External interrupt request flags for external interrupts 9 – 2.

Power-Down Mode

In addition to being able to reduce the power consumption by stopping the clock, there are both idle and power-down modes available. These operate exactly the same as the idle and power-down modes on the 80C51. There is only one difference and that is that it is possible to terminate a power-down condition with either a reset or an external interrupt.

To be able to wake up the part from the power-down state with an external interrupt, both the PD and IDL bits of the PCON register must be set when entering the power-down mode. If only the PD bit is set, the power-down mode will only be terminated by a hardware reset. With both bits set, an interrupt on any of the additional external interrupts, INT2-INT9, will cause the part to wake up. To ensure that the oscillator is stable before the controller restarts, the internal clock will remain inactive for 1536 oscillator periods after the interrupt is detected. After this, the PD flag will be reset, and the part will be in the idle mode, and the interrupt will be handled in the normal way. Figure 2 shows the different oscillator delays associated with the two methods of waking the part up from the power-down mode.

Low Power Consumption

The 8XCL410 is targeted toward low power applications in industrial control, portable instrumentation, intelligent computer peripherals, portable consumer products, and smart cards. Working from a single supply which can vary between 1.8V and 6V, the 8XCL410 requires only a simple voltage regulator. In many cases it can be operated from an unstabilized supply, eliminating altogether the need for a regulator. A typical 8XCL410 device draws 1mA from a 3V supply when running at a 3.5MHz clock frequency. Current consumption in the idle and power-down modes is reduced further to less than 0.5mA and 1μA, respectively.

The 8XCL410 can be operated at clock frequencies up to 16MHz. At these frequencies and a 5V supply voltage, the part will draw current similar to that of a standard 80C51. For the 8XCL410, the reduction in power is a function of both the clock frequency and the supply voltage. Power dissipation is reduced by lowering the clock frequency and/or reducing the supply voltage. A standard Philips 80C51 will operate down to a clock frequency of 0.5MHz and a supply voltage of 4V. The advantage of the low-power 8XCL410 is that it can be run at frequencies as low as 32kHz with the internal oscillator (DC when an external oscillator circuit is used), and that the voltage supply levels can be reduced down to 1.8V. To obtain maximum power reduction, the part can be operated with both reduced clock frequency and reduced voltage supply.

The low voltage operation of the 8XCL410 is due in part to the Philips SACMOS process. This is a self aligned contact CMOS process in which the isolation regions between the contacts and the edge of the isolation have been eliminated. This significantly reduces the size of the die, which in turn reduces the parasitic capacitances and drain resistance. This means that for a given clock speed, parts fabricated in the SACMOS process will require less power, and this is most apparent at low frequencies and voltage supply levels.

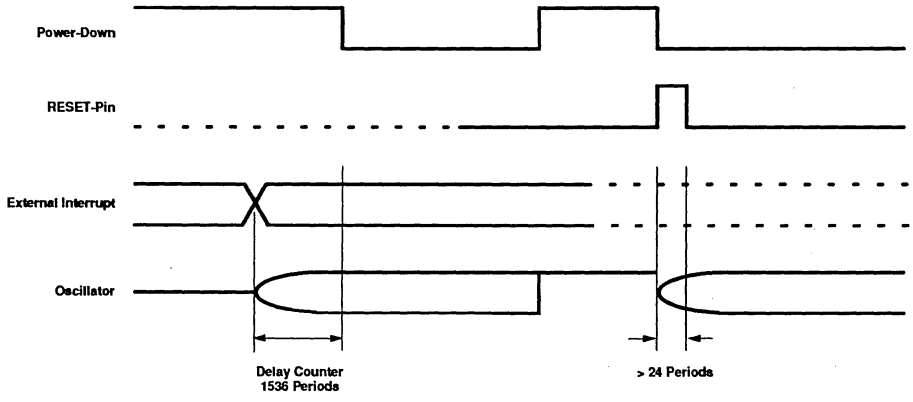


Figure 2. Wake-up Operation

Low voltage/low power single-chip 8-bit microcontroller with I²C

80CL410/83CL410

DESCRIPTION

The 80CL410/83CL410 (hereafter generically referred to as 8XCL410) is manufactured in an advanced CMOS process that allows the part to operate at supply voltages down to 1.8V and oscillator frequencies down to DC. The 8XCL410 has the same instruction set as the 80C51.

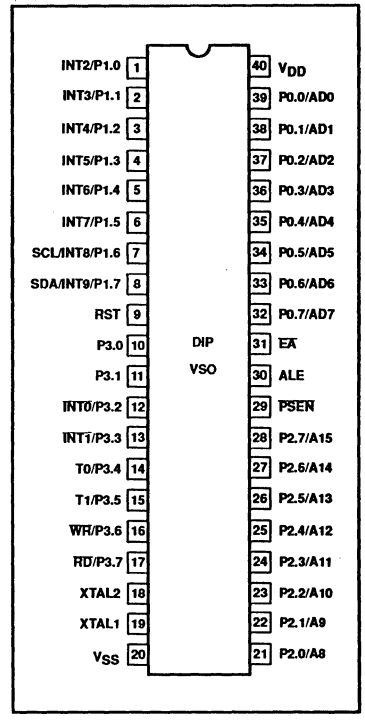
The 8XCL410 features a 4k byte ROM (83CL410), 128 bytes RAM (both ROM and RAM are externally expandable to 64k bytes), four 8-bit ports, two 16-bit timer/counters, an I²C serial interface, a thirteen source, two priority level nested interrupt structure, and on-chip oscillator circuitry suitable for quartz crystal, ceramic resonator, RC, or LC.

The 8XCL410 has two reduced power modes that are the same as those on the standard 80C51. The special reduced power feature of this part is that it can be stopped and then restarted. Running from an external clock source, the clock can be stopped and after a period of time restarted. The 8XCL410 will resume operation from where it was when the code stopped with no loss of internal state, RAM contents, or Special Function Register contents. If the internal oscillator is used the part cannot be stopped and started, but the power-down mode, which can be terminated via an interrupt, can be used to achieve similar power savings and then restart without loss of on-chip RAM and Special Function Register values.

FEATURES

- Supply voltage from 1.8 to 6.0V
- Operating frequency from 32kHz to 12MHz (see Note 1)
- 80C51 based architecture
  - 4k × 8 ROM (64k external)
  - 128 × 8 RAM (64k external)
  - Four 8-bit I/O ports
  - Two 16-bit timer/counters
  - A thirteen-source, two-level, nested priority interrupt structure
  - 10 external interrupts
- Fully static 80C51 CPU
- I²C Serial Interface
- Two power control modes
  - Idle mode
  - Power-down mode – can be terminated by reset or external interrupt
- Wake-up via external interrupts at port 1
- On-chip oscillator (quartz crystal, ceramic resonator, RC, LC)
- Very low power consumption
- Operating temperature range: –40 to +85°C

PIN CONFIGURATION



NOTE:
1. The currently available product is guaranteed up to 12MHz at 4.5V. A 16MHz device will be made available during 1992.

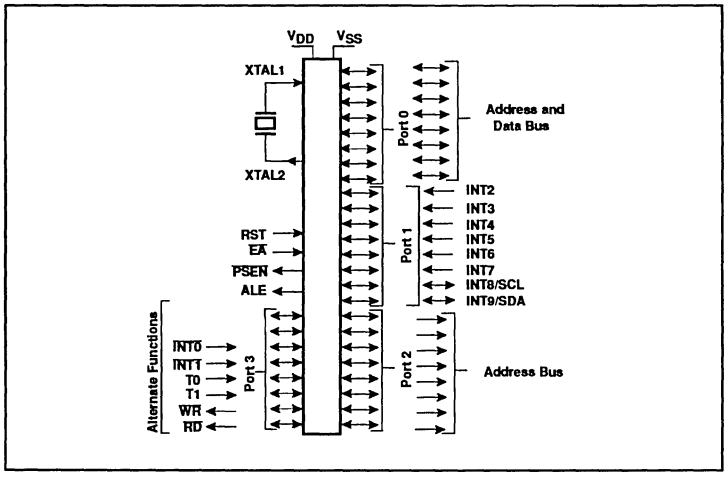
ORDERING CODE

PHILIPS PART ORDER NUMBER PART MARKING		SIGNETICS PART ORDER NUMBER[1]		TEMPERATURE (°C) AND PACKAGE	FREQUENCY
ROMless	ROM	ROMless	ROM		
P80CL410HFP	P83CL410HFP	P80CL410HF N	P83CL410HF N	–40 to +85, plastic DIP	32kHZ to 12MHz
P80CL410HFT	P83CL410HFT	P80CL410HF D	P83CL410HF D	–40 to +85, plastic VSO	32kHZ to 12MHz

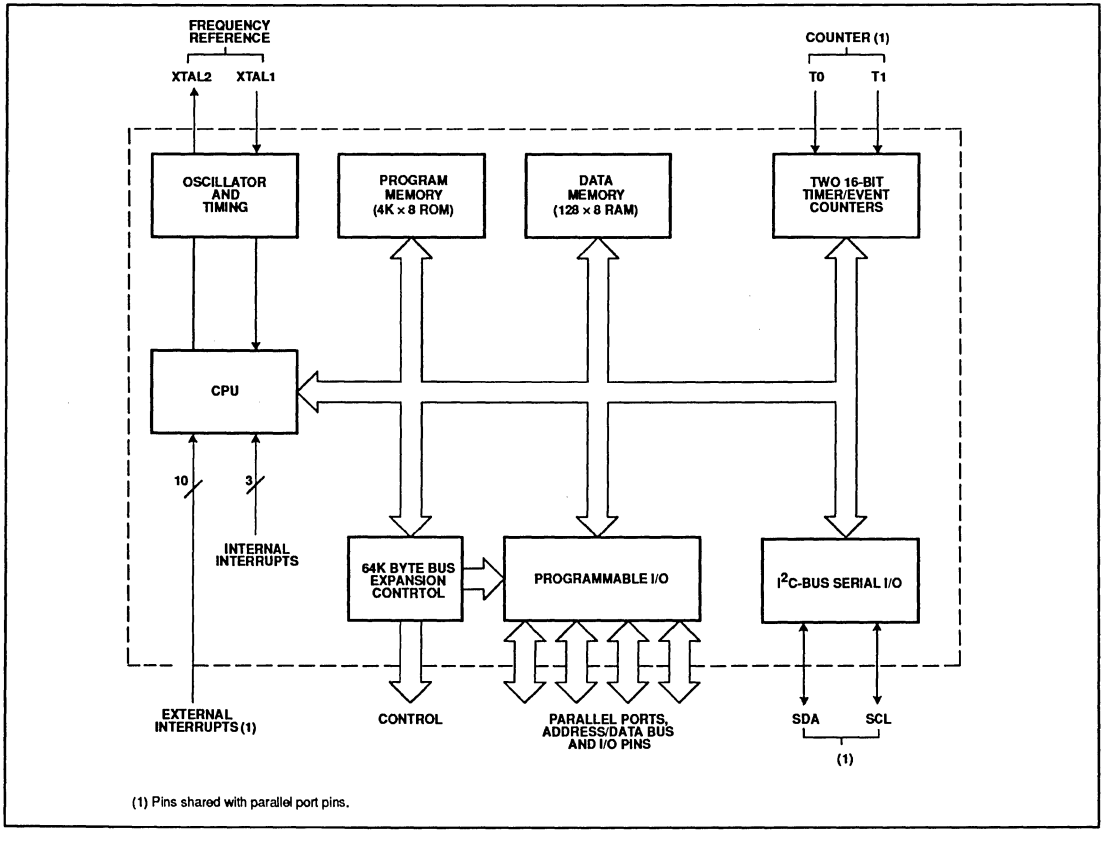
NOTE:
1. Parts ordered by the Signetics part number will be marked with the Philips part marking.

For emulation purposes, the P85CL000 (Piggyback version) with 256 bytes of RAM is recommended.

LOGIC SYMBOL



BLOCK DIAGRAM



(1) Pins shared with parallel port pins.

PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{SS}	20	I	**Ground:** 0V reference.
V_{DD}	40	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s.
P1.0–P1.7	1–8	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Additional functions include:
	7	I/O	**SCL (P1.6):** I²C serial bus clock.
	8	I/O	**SDA (P1.7):** I²C serial bus data.
	1–8	I	**INT2–INT9 (P1.0–P1.7):** Additional external interrupts.
P2.0–P2.7	21–28	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 80C51 family, as listed below:
	12	I	$\overline{\text{INT0}}$ **(P3.2):** External interrupt 0
	13	I	$\overline{\text{INT1}}$ **(P3.3):** External interrupt 1
	14	I	**T0 (P3.4):** Timer 0 external input
	15	I	**T1 (P3.5):** Timer 1 external input
	16	O	$\overline{\text{WR}}$ **(P3.6):** External data memory write strobe
	17	O	$\overline{\text{RD}}$ **(P3.7):** External data memory read strobe
RST	9	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{DD}.
ALE	30	O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory.
$\overline{\text{PSEN}}$	29	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{\text{PSEN}}$ is activated twice each machine cycle, except that two $\overline{\text{PSEN}}$ activations are skipped during each access to external data memory. $\overline{\text{PSEN}}$ is not activated during fetches from internal program memory.
$\overline{\text{EA}}$	31	I	**External Access Enable:** $\overline{\text{EA}}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 1FFFH. If $\overline{\text{EA}}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 0FFFH.
XTAL1	19	I	**Crystal 1:** Input to the inverting oscillator amplifier and input for an external clock source.
XTAL2	18	O	**Crystal 2:** Output from the inverting oscillator amplifier.

PORT OPTIONS

The pins of port 1 (not P1.6/SCL or P1.7/SDA), port 2, and port 3 may be individually configured with one of the following port options (see Figure 1):

Option 1: **Standard Port**—quasi-bidirectional I/O with pull-up. The strong booster pull-up p1 is turned on for two oscillator periods after a 0-to-1 transition in the port latch. See Figure 1(a).

Option 2: **Open Drain**—quasi-bidirectional I/O with n-channel open drain output. Use as an output requires the connection of an external pull-up resistor. See Figure 1(b).

Option 3: **Push-Pull**—output with drive capability in both polarities. Under this option, pins can only be used as outputs. See Figure 1(c).

The definition of port options for port 0 is slightly different.

Two cases have to be examined. First, accesses to external memory ($\overline{EA}$ = 0 or access above the built-in memory boundary), and second, I/O accesses.

External Memory Accesses

Option 1: True 0 and 1 are written as address to the external memory (strong pull-up is used).

Option 2: An external pull-up resistor is needed for external accesses.

Option 3: True 0 and 1 are written as address to the external memory (strong pull-up is used).

I/O Accesses

Option 1: When writing a 1 to the port latch, the strong pull-up p1 will be on for two oscillator periods. No weak pull-up exists. Without an external pull-up, this option can be used as a high-impedance input.

Option 2: Open drain—quasi-bidirectional I/O with n-channel open drain output. Use as an output requires the connection of an external pull-up resistor. See Figure 1(c).

Option 3: Push-Pull—output with drive capability in both polarities. Under this option, pins can only be used as outputs.

Individual mask selection of the post-reset state is available on any of the above pins. Make your selection by appending "S" or "R" to option 1, 2, or 3 above (e.g., 1S for a standard I/O to be set after RESET or 2R for an open-drain I/O to be reset after RESET.

Option S: **Set**—after reset, this pin will be initialized High.

Option R: **Reset**—after reset, this pin will be initialized Low.

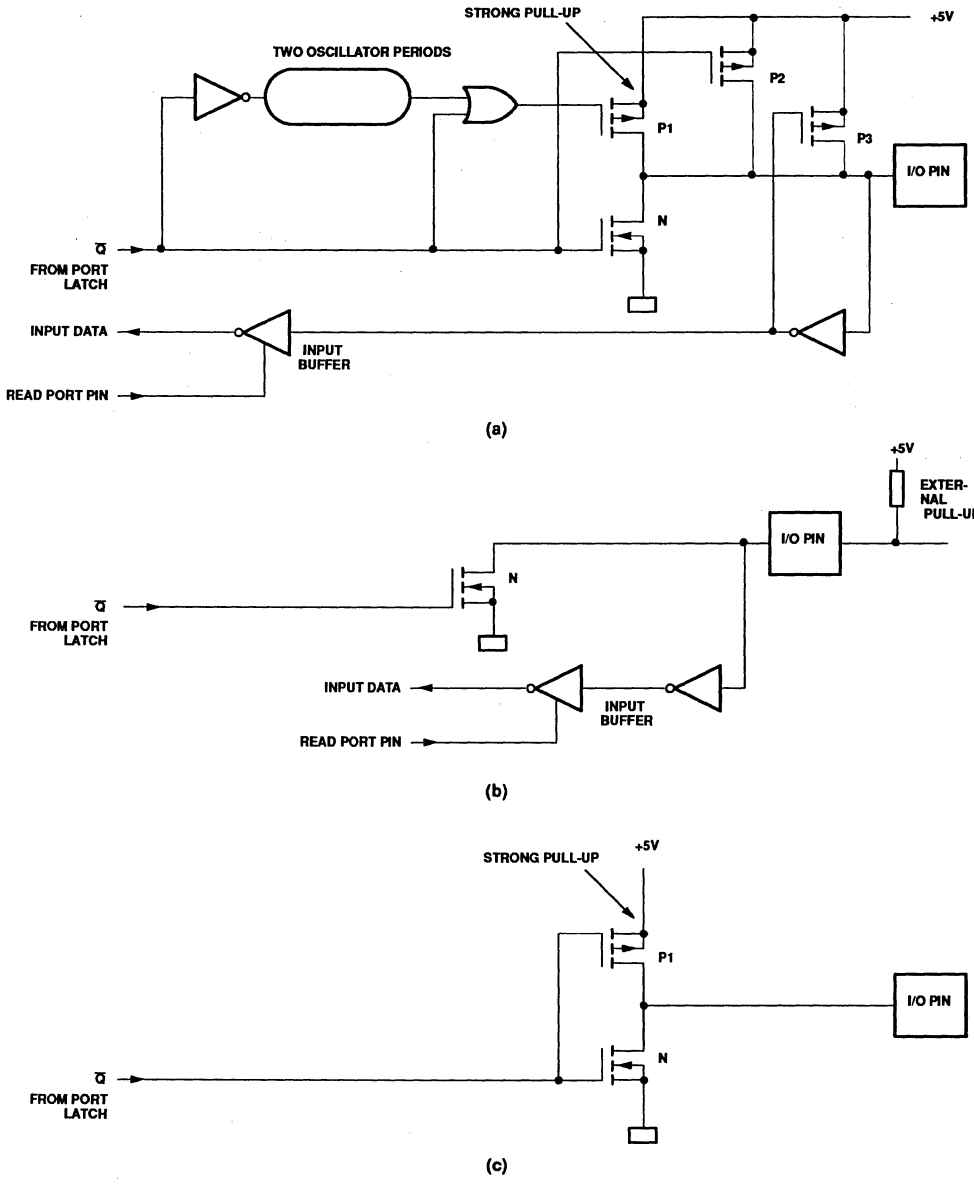


Figure 1. Ports

POWER-DOWN MODE

The instruction setting PCON.1 is the last executed prior to going into the power-down mode. In power-down mode, the oscillator is stopped. The contents of the the on-chip RAM and SFRs are preserved. The port pins output the values held by their respective SFRs. ALE and $\overline{PSEN}$ are held low. Power-down operates in wake-up mode and reset mode.

In the power-down mode, V_{DD} may be reduced to minimize power consumption. However, the supply voltage must not be reduced until the power-down mode is active, and must be restored before the hardware reset is applied and frees the oscillator. Reset must be held active until the oscillator has restarted and stabilized.

Wake-Up Mode

Setting both PD and IDL flags in the PCON register forces the controller into the power-down mode. Setting both flags enable the controller to be woken-up from the power-down mode with either the external interrupts INT2–INT9, or a reset operation.

An external interrupt INT2–INT9 at port 1 releases both the oscillator and the delay counter. To ensure that the oscillator is stable before the controller restarts, the internal clock will remain inactive for 1536 oscillator periods after the interrupt is detected. After this, the PD flag will be reset, the controller is now in the Idle mode and the interrupt will be handled in the normal way.

Reset Mode

Setting only the PD bit in the PCON register again forces the controller into the power-down mode, but in this case it can only be restored to normal operation with a direct reset operation.

IDLE MODE

The instruction that sets PCON.0 is the last instruction executed before going into idle mode. In idle mode, the internal clock is stopped for the CPU, but not for the interrupt, timer, and serial port functions. The CPU status is preserved along with the stack pointer, program counter, program status word and accumulator. The RAM and all other registers maintain their data during idle mode. The port pins retain the logical states they held at idle mode activation. ALE and $\overline{PSEN}$ hold at the logic high level.

There are two methods used to terminate the idle mode. Activation of any interrupt will cause PCON to be cleared by hardware; terminating idle mode. The interrupt is serviced, and following the instruction RETI, the next instruction to be executed will be the one following the instruction that put the device in the the idle mode.

Flag bits GF0 and GF1 can be used to determine whether the interrupt was received during normal execution or idle mode. For example, the instruction that writes to PCON.0 can also set or clear one or both flag bits. When idle mode is terminated by an interrupt, the service routine can examine the status of the flag bits.

The second method of terminating the idle mode is with an external hardware reset. Since the oscillator is still running, the hardware reset is required to be active for only two machine cycles to complete the reset operation. Reset redefines all SFRs, but does not affect the on-chip RAM.

The status of the external pins during idle and power-down mode is shown in Table 1. If the power-down mode is activated while accessing external memory, port data held in the special function register P2 is restored to port 2. If the data is a logic 1, the port pin is held high during the power-down mode.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Floating	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Floating	Data	Data	Data

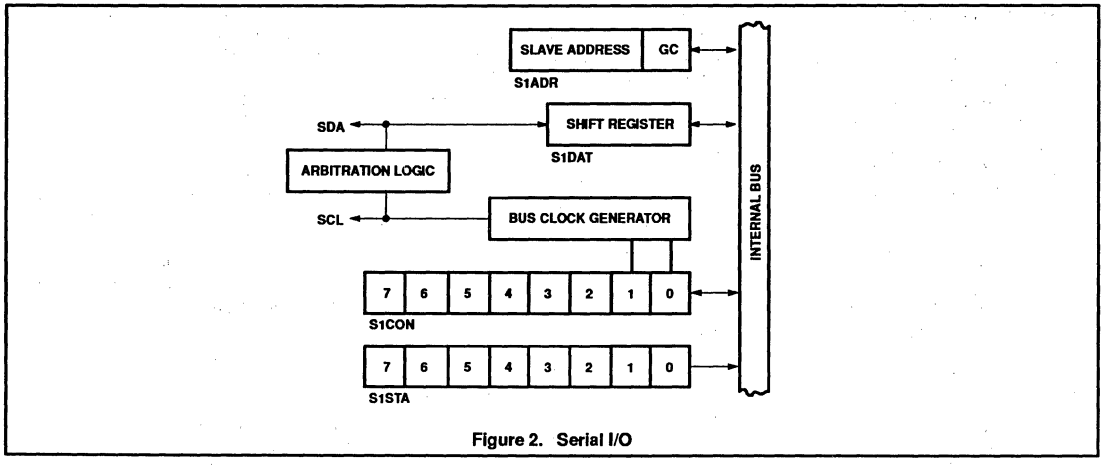


Figure 2. Serial I/O

I²C-BUS SERIAL I/O

The serial port supports the twin line I²C-bus. The I²C-bus consists of a data line (SDA) and a clock line (SCL). These lines also function as I/O port lines P1.7 and P1.6 respectively. The system is unique because data transport, clock generation, address recognition and bus control arbitration are all controlled by hardware. The I²C-bus serial I/O has complete autonomy in byte handling and operates in four modes:

- Master transmitter
- Master receiver
- Slave transmitter
- Slave receiver

These functions are controlled by the S1CON register. S1STA is the status register whose contents may also be used as a vector to various service routines. S1DAT is the data shift register and S1ADR the slave address register. Slave address recognition is performed by hardware.

S1CON (D8H)
Serial control register

—	ENS1	STA	STO	SI	AA	CR1	CR0

CR0, CR1 These two bits determine the serial clock frequency when SIO is in a master mode.

AA Assert acknowledge bit. When the AA flag is set, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when:
- own slave address is received
- general call address is received (S1ADR.0 = 1)
- data byte received while device is programmed as master
- data byte received while device is selected slave

Withe AA = 0, no acknowledge will be returned. Consequently, no interrupt is requested when the "own slave address" or general call address is received.

SI SIO interrupt flag. When the SI flag is set, an acknowledge is returned after any one of the following conditions:
- a start condition is generated in master mode
- own slave address received during AA = 1
- general call address received while S1ADR.0 and AA = 1
- data byte received or transmitted in master mode (even if arbitration is lost)
- data byte received or transmitted as selected slave
- stop or start condition received as selected slave receiver or transmitter

STO STOP flag. With this bit set while in master mode, a STOP condition is generated. When a STOP condition is detected on the bus, the SIO hardware clears the STO flag. In the slave mode, the STO flag may also be set to recover from an error condition. In this case, no STOP condition is transmitted to the I²C-bus. However, the SIO hardware behaves as if a STOP condition has been received and releases SDA and SCL. The SIO then switches to the "not addressed" slave receiver mode. The STO flag is automatically cleared by hardware.

STA START flag. When the STA bit is set in slave mode, the SIO hardware checks the status of the I²C-bus and generates a START condition if the bus is free. If STA is set while the SIO is in master mode, SIO transmits a repeated START condition.

ENS1 When ENS1 = 0, the SIO is disabled. The SDA and SCL outputs are in a high-impedance state; P1.6 and P1.7 function as open drain ports.

When ENS1 = 1, the SIO is enabled. The P1.6 and P1.7 port latches must be set to logic 1.

S1STA (D9H)
Status register

SC4	SC3	SC2	SC1	SC0	0	0	0

S1STA is an 8-bit read-only special function register. S1STA.3–S1STA.7 hold a status code. S1STA.0–S1STA.2 are held LOW. The contents of S1STA may be used as a vector to a service routine. This optimizes response time of the software and consequently that of the I²C-bus.

The following is a list of the status codes:

Abbreviations used:

SLA: 7-bit slave address
R: Read bit
W: Write bit
$\overline{ACK}$: Acknowledgement (acknowledge bit = 0)
ACK: Not Acknowledge (acknowledge bit = 1)
DATA: 8-bit byte to or from the I²C-bus
MST: Master
SLV: Slave
TRX: Transmitter
REC: Receiver

MST/TRX mode
S1STA value

08H – a START condition has been transmitted
10H – a repeated START condition has been transmitted
18H – SLA and W have been transmitted, ACK received
20H – SLA and W have been transmitted, $\overline{ACK}$ received
28H – DATA of S1DAT has been transmitted, ACK received
30H – DATA of S1DAT has been transmitted, $\overline{ACK}$ received
38H – Arbitration lost in SLA, R/W or DATA

MST/REC mode
S1STA value

38H – Arbitration lost while returning $\overline{ACK}$
40H – SLA and R have been transmitted, ACK received
48H – SLA and R have been transmitted, $\overline{ACK}$ received
50H – DATA has been received, ACK returned
58H – DATA has been received, $\overline{ACK}$ returned

SLV/REC mode
S1STA value

60H – Own SLA and W have been received, ACK returned
68H – Arbitration lost in SLA, R/W as MST. Own SLA and W have been received, ACK returned
70H – General CALL has been received, ACK returned
78H – Arbitration lost in SLA, R/W as MST. General CALL has been received
80H – Previously addressed with own SLA. DATA byte received, ACK returned
88H – Previously addressed with own SLA. DATA byte received, $\overline{ACK}$ returned
90H – Previously addressed with general CALL. DATA byte has been received, ACK has been returned
98H – Previously addressed with general CALL. DATA byte has been received, $\overline{ACK}$ has been returned
A0H – A STOP condition or repeated START condition has been received while still addressed as SLV/REC or SLV/TRX

SLV/TRX mode
S1STA value

A8H – Own SLA and R have been received, ACK returned
B0H – Arbitration lost in SLA, R/W as MST. Own SLA and R have been received, ACK returned
B8H – DATA byte has been transmitted, ACK received
C0H – DATA byte has been transmitted, $\overline{ACK}$ received
C8H – Last DATA byte has been transmitted (AA = logic 0), ACK received

Miscellaneous
S1STA value

00H – Bus error during MST mode or selected SLV mode, due to an erroneous START or STOP condition

S1DAT (DAH)
Data Shift Register

7	6	5	4	3	2	1	0

Data shift register S1DAT
This register contains the serial data to be transmitted or data that has just been received. Bit 7 is transmitted or received first, i.e., data is shifted from left to right.

S1ADR (DBH)
Slave Address Register

7	6	5	4	3	2	1	0

S1ADR.0, GC: 0 = general CALL address is not recognized
1 = general CALL address is recognized

S1ADR.7-1: own slave address

This 8-bit register may be loaded with the 7-bit slave address, to which the controller will respond when programmed as a slave receiver/transmitter. The LSB bit (GC) is used to determine whether the general CALL address is recognized.

INTERRUPT SYSTEM

External events and the real-time-driven on-chip peripherals require service by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution, a multiple-source, two-priority level, nested interrupt system is provided. The 8XCL410 acknowledges interrupt requests from thirteen sources, as follows:

- INT0 and INT1
- Timer 0 and timer 1
- I²C-bus serial I/O interrupt
- INT2 to INT9 (port 1)

Each interrupt vectors to a separate location in program memory for its service routine. Each source can be individually enabled or disabled by corresponding bits in the internal enable registers (IEN0, IEN1) The priority level is selected via the interrupt priority register (IP0, IP1). All enabled sources can be globally disabled or enabled.

External Interrupts INT2–INT9

Port 1 lines serve an alternative purpose as eight additional interrupts INT2–INT9. When enabled, each of these lines can "wake-up" the device from power-down mode. Using the IX1 register, each pin may be initialized to either active high or low. IRQ1 is the interrupt request flag register. Each flag, if the interrupt is enabled, will be set on an interrupt request but it must be cleared by software.

IEN1 (E8H)
Interrupt enable register

7	6	5	4	3	2	1	0
EX9	EX8	EX7	EX6	EX5	EX4	EX3	EX2

Bit	Symbol	Function
IEN1.7	EX9	Enable external interrupt 9
IEN1.6	EX8	Enable external interrupt 8
IEN1.5	EX7	Enable external interrupt 7
IEN1.4	EX6	Enable external interrupt 6
IEN1.3	EX5	Enable external interrupt 5
IEN1.2	EX4	Enable external interrupt 4
IEN1.1	EX3	Enable external interrupt 3
IEN1.0	EX2	Enable external interrupt 2

where 0 = interrupt disabled
1 = interrupt enabled

IP1 (F8H)
Interrupt priority register

7	6	5	4	3	2	1	0
PX9	PX8	PX7	PX6	PX5	PX4	PX3	PX2

Bit	Symbol	Function
IP1.7	PX9	External interrupt 9 priority level
IP1.6	PX8	External interrupt 8 priority level
IP1.5	PX7	External interrupt 7 priority level
IP1.4	PX6	External interrupt 6 priority level
IP1.3	PX5	External interrupt 5 priority level
IP1.2	PX4	External interrupt 4 priority level
IP1.1	PX3	External interrupt 3 priority level
IP1.0	PX2	External interrupt 2 priority level

Interrupt priority is as follows:
0 – low priority
1 – high priority

IX1 (E9H)
Interrupt polarity register

7	6	5	4	3	2	1	0
IL9	IL8	IL7	IL6	IL5	IL4	IL3	IL2

Bit	Symbol	Function
IX1.7	IL9	External interrupt 9 polarity level
IX1.6	IL8	External interrupt 8 polarity level
IX1.5	IL7	External interrupt 7 polarity level
IX1.4	IL6	External interrupt 6 polarity level
IX1.3	IL5	External interrupt 5 polarity level
IX1.2	IL4	External interrupt 4 polarity level
IX1.1	IL3	External interrupt 3 polarity level
IX1.0	IL2	External interrupt 2 polarity level

Writing either a "1" or "0" to an IX1 register bit sets the priority level of the corresponding external interrupt to active High or Low, respectively.

IRQ1 (C0H)
Interrupt request flag register

7	6	5	4	3	2	1	0
IQ9	IQ8	IQ7	IQ6	IQ5	IQ4	IQ3	IQ2

Bit	Symbol	Function
IRQ1.7	IQ9	External interrupt 9 request flag
IRQ1.6	IQ8	External interrupt 8 request flag
IRQ1.5	IQ7	External interrupt 7 request flag
IRQ1.4	IQ6	External interrupt 6 request flag
IRQ1.3	IQ5	External interrupt 5 request flag
IRQ1.2	IQ4	External interrupt 4 request flag
IRQ1.1	IQ3	External interrupt 3 request flag
IRQ1.0	IQ2	External interrupt 2 request flag

Priority	Vector	Source
X0 (highest)	0003H	External 0
S1	002BH	I²C port
X5	0053H	External 5
T0	000BH	Timer 0
X6	005BH	External 6
X1	0013H	External 1
X2	003BH	External 2
X7	0063H	External 7
T1	001BH	Timer 1
X3	0043H	External 3
X8	006BH	External 8
X4	004BH	External 4
X9 (lowest)	0073H	External 9

Register	Function	SFR Address
IX1	Interrupt polarity register	E9H
IRQ1	Interrupt request flag register	C0H
IEN0	Interrupt enable register	A8H
IEN1	Interrupt enable register (INT2–INT9)	E8H
IP0	Interrupt priority register	B8H
IP1	Interrupt priority register (INT2–INT9)	F8H

OSCILLATOR CIRCUITRY

The on-chip oscillator circuitry of the 8XCL410 is a single stage inverting amplifier biased by an internal feedback resistor. (See Figure 3.) The oscillator can be operated with a quartz crystal, ceramic resonator, LC network or RC network. See Figure 4 for different configurations. When ordering parts, it is necessary to specify an oscillator option. The options are: RC when an RC network will be used, OSC 2 for oscillator operation below 4MHz, OSC 3 for oscillator operation from 4MHz to 10MHz, OSC 4 for oscillator operation above 10MHz, and 32kHz if 32kHz to 400kHz operation is desired.

For operation as a standard quartz oscillator, no external components are needed (except at 32KHz). When using external capacitors, ceramic resonators, coils, and RC networks to drive the oscillator, five different configurations are supported (see Figure 4 and Table 2).

In the power-down mode the oscillator is stopped and XTAL1 is pulled high. The oscillator inverter is switched off to ensure no current will flow. To drive the device with an external clock source, apply the external clock signal to XTAL1, and leave XTAL2 to float, as shown in Figure 4(f). There are no requirements on the duty cycle of the external clock, since the input to the internal clocking circuitry is split using a flip-flop.

The following options are provided for optimum on-chip oscillator performance. Please state option when ordering:

32kHz: Figure 4(c). An option for 32kHz clock applications with external trimmer for frequency adjustment.

A 4.7MΩ bias resistor must be connected in parallel with the crystal.

Osc.2: Figure 4(e). An option for low-power, low-frequency operations using LC components or quartz.

Osc.3: An option for medium frequency range applications.

Osc.4: An option for high frequency range applications.

RC: Figure 4(g). An option for an RC oscillator.

The equivalent circuit data of the internal oscillator compares with that of matched crystals.

The externally adjustable RC oscillator has a frequency range from 100kHz to 500kHz. (See Figure 6.)

Power-on Reset

The 8XCL410 contains on-chip circuitry which switch the port pins to the customer-defined logic level as soon as V_{DD} exceeds 1.3V. (See Figures 7 and 8.) As soon as the minimum supply voltage is reached, the oscillator will start up. However, to ensure that the oscillator is stable before the controller starts, the clock signals are gated away from the CPU for a further 1536 oscillator periods.

An hysteresis of approximately 100mV at a typical power-on switching level of 1.3V will ensure correct operation.

An automatic reset can be obtained at power-on by connecting the RST pin to V_{DD} via a 10µF capacitor. At power-on, the voltage on the RST pin is equal to V_{DD} minus the capacitor voltage, and decreases from V_{DD} as the capacitor discharges through the internal resistor R_{RST} to ground. The larger the capacitor, the more slowly V_{RST} decreases. V_{RST} must remain above the lower threshold of the Schmitt trigger long enough to effect a complete reset. The time required is the oscillator start-up time, plus 2 machine cycles.

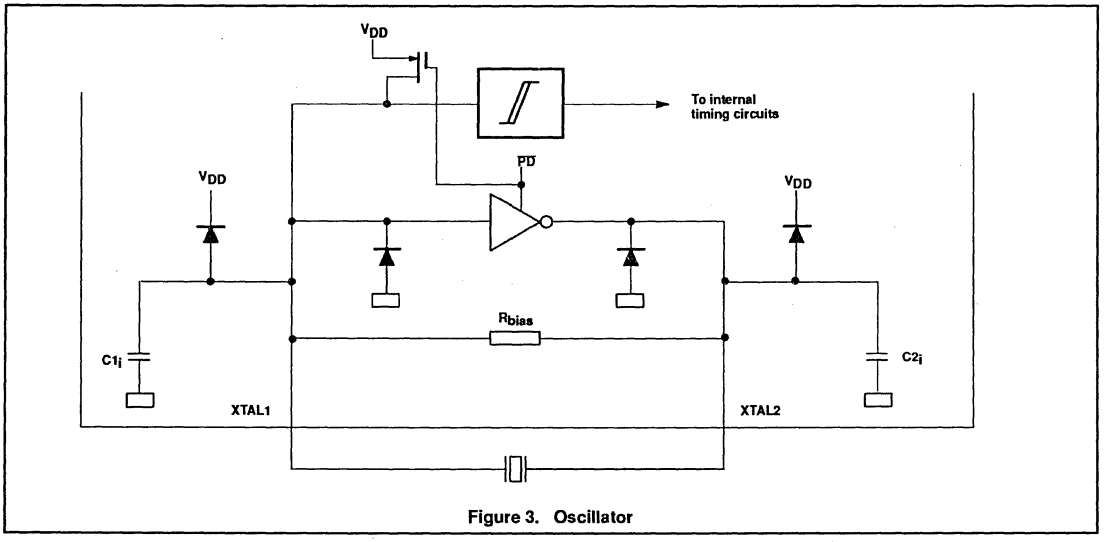


Figure 3. Oscillator

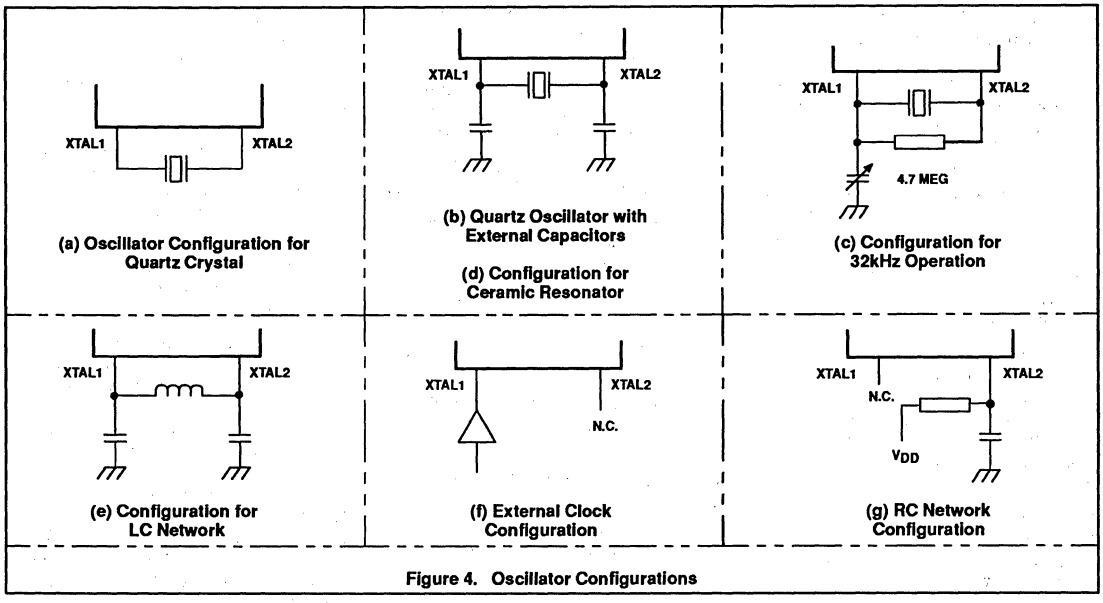


Figure 4. Oscillator Configurations

Table 2. Oscillator Type Selection Guide

RESONATOR	f (MHZ)	OPTION	C1 EXT. MIN	C1 EXT. MAX	C2 EXT. MIN	C2 EXT. MAX	MAXIMUM RESONATOR SERIES RESISTANCE
Quartz	0.032	32kHz	5	15	0	0	15kΩ[1]
Quartz	1.0	Osc.2	0	30	0	30	600Ω
Quartz	3.58	Osc.2	0	15	0	15	100Ω
Quartz	4.0	Osc.2	0	20	0	20	75Ω
Quartz	6.0	Osc.3	0	10	0	10	60Ω
Quartz	10.0	Osc.4	0	15	0	15	60Ω
Quartz	12.0	Osc.4	0	10	0	10	40Ω
Quartz	16.0	Osc.4	0	15	0	15	20Ω
PXE	0.455	Osc.2	40	50	40	50	10Ω
PXE	1.0	Osc.2	15	50	15	50	100Ω
PXE	3.58	Osc.2	0	40	0	40	10Ω
PXE	4.0	Osc.2	0	40	0	40	10Ω
PXE	6.0	Osc.2	0	20	0	20	5Ω
PXE	10.0	Osc.3	0	15	0	15	6Ω
PXE	12.0	Osc.4	10	40	10	40	6Ω
LC		Osc.2	20	90	20	90	10μH = 1Ω 100μH = 5Ω 1mH = 75Ω

NOTE:
1. 32kHz quartz crystals with a series resistance higher than 15kΩ will reduce the guaranteed supply voltage range to 2.5 to 3.5V.

Table 3. Oscillator Equivalent Circuit Parameters (see Figure 5)

PARAMETER	OPTION	SYMBOL	CONDITION	MIN	TYP	MAX	UNIT
Transconductance	32kHz	g_m	T = +25°C; V_{DD} = 4.5V	–	15	–	µs
	Osc.2	g_m	T = +25°C; V_{DD} = 4.5V	–	600	–	µs
	Osc.3	g_m	T = +25°C; V_{DD} = 4.5V	–	1500	–	µs
	Osc.4	g_m	T = +25°C; V_{DD} = 4.5V	–	4000	–	µs
Input capacitance	32kHz	$c1_i$		–	3.0	–	pF
	Osc.2	$c1_i$		–	8.0	–	pF
	Osc.3	$c1_i$		–	8.0	–	pF
	Osc.4	$c1_i$		–	8.0	–	pF
Output capacitance	32kHz	$c2_i$		–	23.0	–	pF
	Osc.2	$c2_i$		–	8.0	–	pF
	Osc.3	$c2_i$		–	8.0	–	pF
	Osc.4	$c2_i$		–	8.0	–	pF
Output resistance	32kHz	R2		–	3800	–	kΩ
	Osc.2	R2		–	65	–	kΩ
	Osc.3	R2		–	18	–	kΩ
	Osc.4	R2		–	5.0	–	kΩ

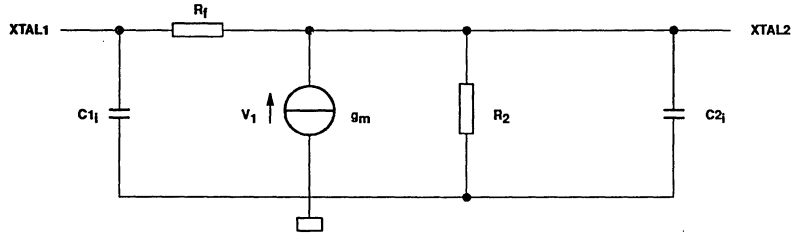


Figure 5. Equivalent Circuit Diagram

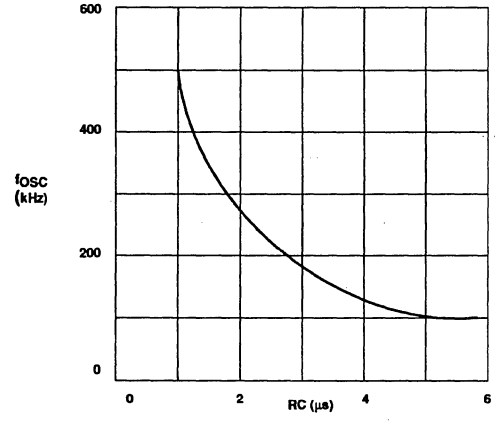


Figure 6. Frequency as a Function of RC

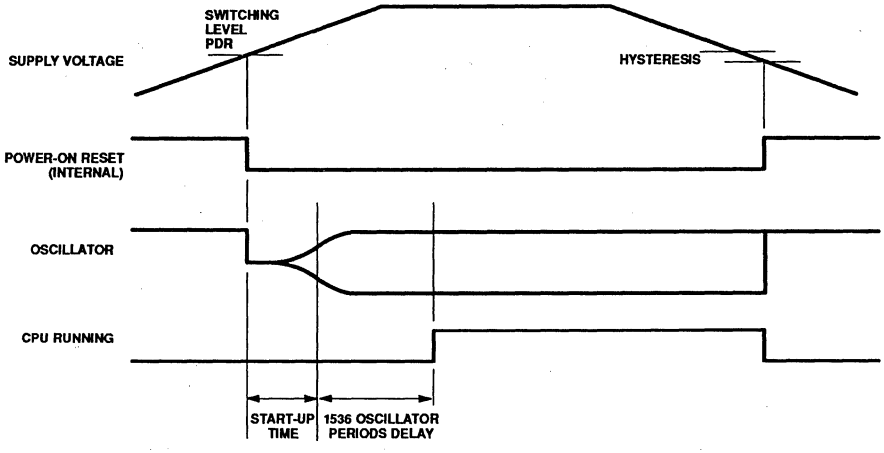


Figure 7. Power-on Reset Switching Level

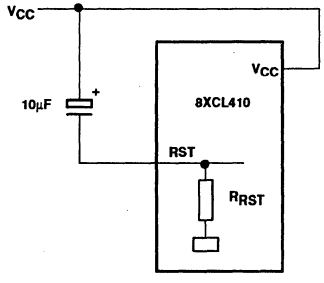


Figure 8. Recommended Power-on Reset Circuitry

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Supply voltage	−0.5 to +6.5	V
All input voltages	−0.5 to V_{DD} +0.5	V
DC current into any input or output	5	mA
Total power dissipation	300	mW
Storage temperature range	−65 to +150	°C
Operating ambient temperature range	−40 to +85	°C
Operating junction temperature	125	°C

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

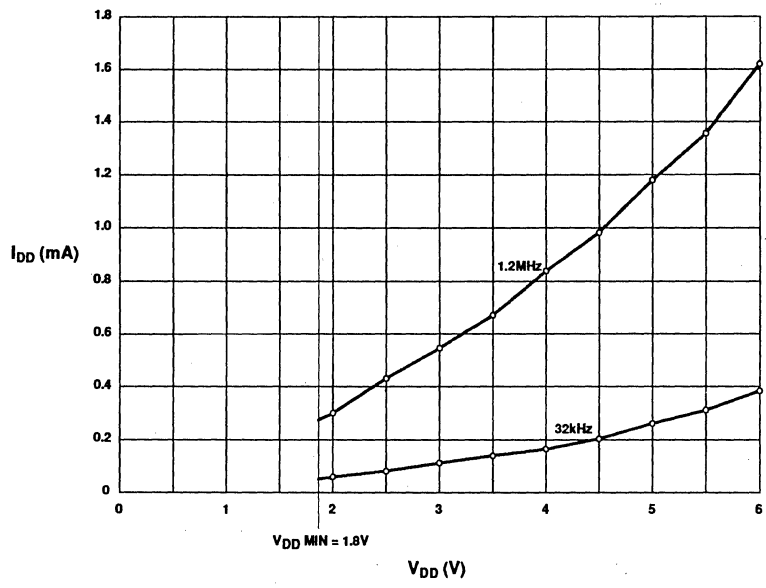
DC ELECTRICAL CHARACTERISTICS

T_{amb} = –40°C to +85°C, V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{DD}	Supply voltage	f_{CLK} (see Figure 12)	1.8	6.0	V
	RAM retention voltage in power-down mode		1.0	—	V
I_{DD}	Power supply current:				
	Operating[1]				
	OSC 1 option	f_{CLK} = 32kHz, V_{DD} = 1.8V, T_{amb} = +25°C	—	50	µA
	OSC 2 option	f_{CLK} = 3.58MHz, V_{DD} = 3V	—	2.5	mA
	OSC 2 option	f_{CLK} = 10MHz, V_{DD} = 5V	—	14	mA
	OSC 3 option	f_{CLK} = 16MHz, V_{DD} = 5V	—	18	mA
	OSC 4 option	f_{CLK} = 16MHz, V_{DD} = 5V	—	22	mA
	Idle mode[2]				
	OSC 1 option	f_{CLK} = 32kHz, V_{DD} = 1.8V, T_{amb} = +25°C	—	25	µA
	OSC 2 option	f_{CLK} = 3.58MHz, V_{DD} = 3V	—	1.0	mA
	OSC 2 option	f_{CLK} = 10MHz, V_{DD} = 5V	—	5.0	mA
	OSC 3 option	f_{CLK} = 16MHz, V_{DD} = 5V	—	7.5	mA
	OSC 4 option	f_{CLK} = 16MHz, V_{DD} = 5V	—	9.0	mA
	Power-down mode[3]	V_{DD} = 1.8V, T_{amb} = +25°C	—	10	µA
V_{IL}	Input low voltage		V_{SS}	0.3V_{DD}	V
V_{IH}	Input high voltage		0.7V_{DD}	V_{DD}	V
I_{OL}	Output sink current, except SDA, SCL	V_{DD} = 5V, V_{OL} = 0.4V	1.6		mA
		V_{DD} = 2.5V, V_{OL} = 0.4V	0.7		mA
I_{OL1}	Output sink current, SDA, SCL	V_{DD} = 5V, V_{OL} = 0.4V	3.0		mA
I_{OH}	Output source current (push-pull options only)	V_{DD} = 5V, V_{OH} = V_{DD} – 0.4V	1.6		mA
		V_{DD} = 2.5V, V_{OH} = V_{DD} – 0.4V	0.7		mA
I_{IL}	Logical 0 input current, ports 1, 2, 3	V_{DD} = 5V, V_{IN} = 0.4V		–100	µA
		V_{DD} = 2.5V, V_{IN} = 0.4V		–50	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3	V_{DD} = 5V, V_{IN} = V_{DD}/2		–1.0	mA
		V_{DD} = 2.5V, V_{IN} = V_{DD}/2		–500	µA
I_{LI}	Input leakage current, port 0, $\overline{EA}$, SCL, SDA	V_{SS} < V_I < V_{DD}		±10	µA
R_{RST}	Internal reset pull-down resistor		10	200	kΩ

NOTES:

1. The operating supply current is measured with all output pins disconnected; XTAL1 driven with t_r = t_f = 10ns; V_{IL} = V_{SS}, V_{IH} = V_{DD}; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = V_{DD}; all open drain outputs connected to V_{SS}.
2. The idle supply current is measured with all output pins disconnected; XTAL1 driven with t_r = t_f = 10ns; V_{IL} = V_{SS}, V_{IH} = V_{DD}; XTAL2 not connected; $\overline{EA}$ = Port 0 = V_{DD}; RST = V_{SS}; all open drain outputs connected to V_{SS}.
3. The power-down current is measured with all output pins disconnected; XTAL1 not connected; $\overline{EA}$ = port 0 = V_{DD}; RST = V_{SS}; all open-drain outputs connected to V_{SS}.
4. The RC-oscillator is not implemented in this version.
5. Circuits with option "no power-on reset" are tested at V_{DDMIN} = 1.8V, with option POR = 1.3V at V_{DDMIN} = 2.5V.



Typical Operating Current Versus Supply and Frequency (32kHz–1.2MHz) at +25°C

AC ELECTRICAL CHARACTERISTICS

T_{amb} = –40°C to +85°C, V_{SS} = 0V[1, 2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
Program Memory							
$1/t_{CLCL}$		Oscillator frequency			0	20	MHz
t_{LL}	9	ALE pulse width	127		$2t_{CLCL}$–40		ns
t_{AL}	9	Address valid to ALE low	43		t_{CLCL}–40		ns
t_{LA}	9	Address hold after ALE low	48		t_{CLCL}–35		ns
t_{LIV}	9	ALE low to valid instruction in		233		$4t_{CLCL}$–100	ns
t_{LC}	9	ALE low to $\overline{PSEN}$ low	58		t_{CLCL}–25		ns
t_{CC}	9	$\overline{PSEN}$ pulse width	215		$3t_{CLCL}$–35		ns
t_{CIV}	9	$\overline{PSEN}$ low to valid instruction in		125		$3t_{CLCL}$–125	ns
t_{CI}	9	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{CIF}	9	Input instruction float after $\overline{PSEN}$		63		t_{CLCL}–20	ns
t_{AVI}	9	Address to valid instruction in		302		$5t_{CLCL}$–115	ns
t_{AFC}	9	$\overline{PSEN}$ low to address float	0		0		ns
Data Memory							
t_{RR}	10	$\overline{RD}$ pulse width	400		$6t_{CLCL}$–100		ns
t_{WW}	11	$\overline{WR}$ pulse width	400		$6t_{CLCL}$–100		ns
t_{LA}	10, 11	Address hold time after ALE	48	–	t_{CLCL}–35	–	ns
t_{RD}	10	$\overline{RD}$ low to valid data in		250		$5t_{CLCL}$–165	ns
t_{DFR}	10	Data float after $\overline{RD}$		97		$2t_{CLCL}$–70	ns
t_{LD}	10	ALE low to valid data in		517		$8t_{CLCL}$–150	ns
t_{AD}	10	Address to valid data in		585		$9t_{CLCL}$–165	ns
t_{LW}	10, 11	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}$–50	$3t_{CLCL}$+50	ns
t_{AW}	10, 11	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}$–130		ns
t_{DWX}	11	Data valid to $\overline{WR}$ transition	23		t_{CLCL}–60		ns
t_{DW}	10	Data valid to $\overline{WR}$	433	–	$7t_{CLCL}$–150	–	ns
t_{WD}	11	Data hold after $\overline{WR}$	33		t_{CLCL}–50		ns
t_{AFR}	10	$\overline{RD}$ low to address float[3]		12		12	ns
t_{WHLH}	10, 11	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	t_{CLCL}–40	t_{CLCL}+40	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 50pF, load capacitance for all other outputs = 40pF.
3. Interfacing the 8XCL410 to devices with float time up to 75ns is permitted. This limited bus connection will not cause damage to port 0 drivers.

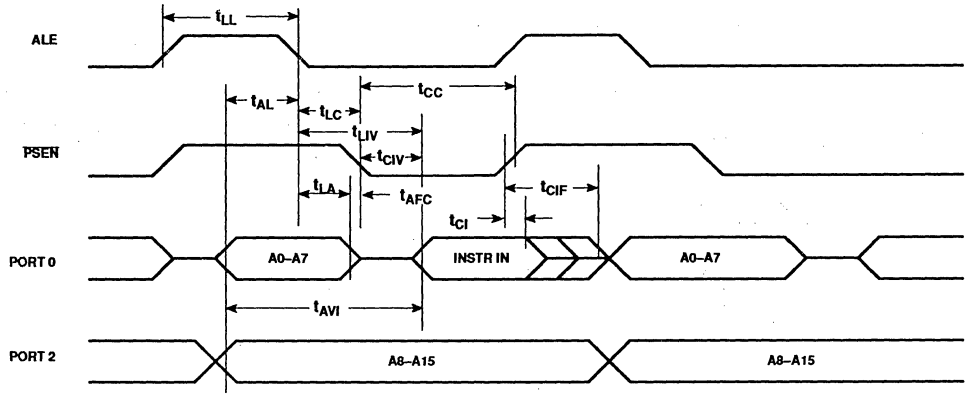


Figure 9. External Program Memory Read Cycle

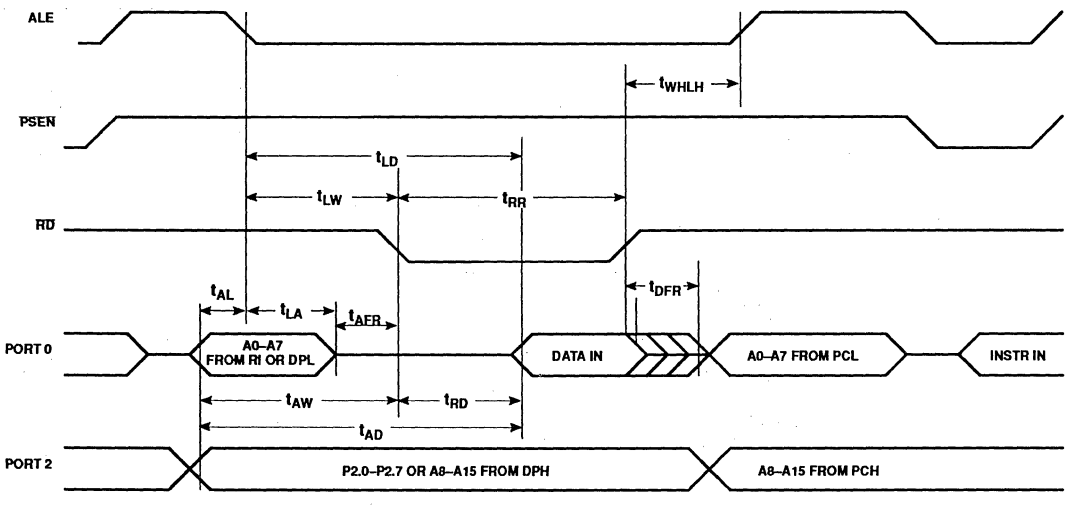


Figure 10. External Data Memory Read Cycle

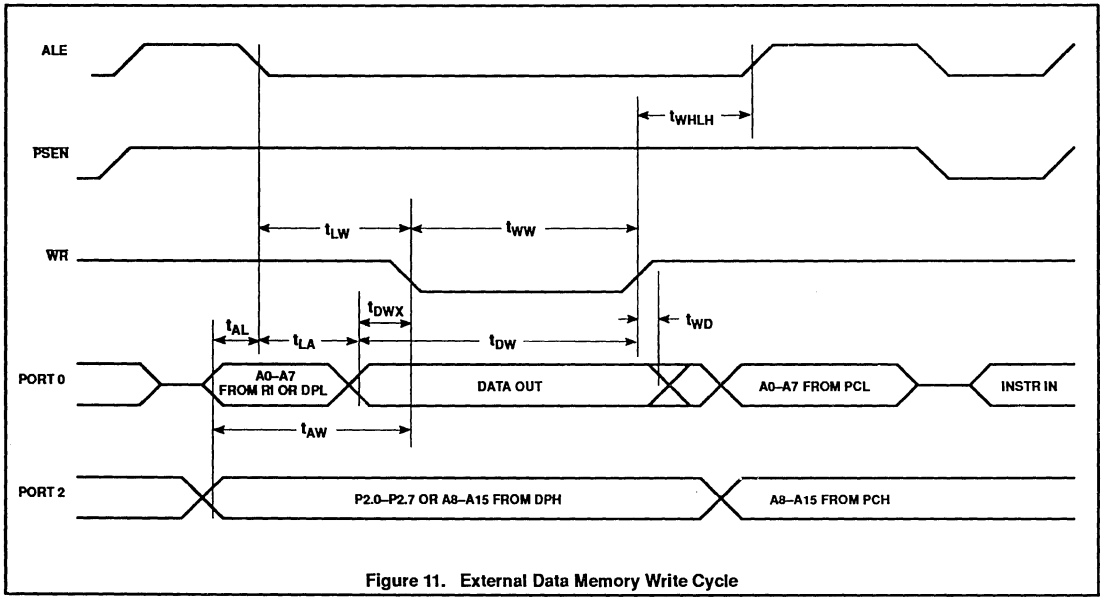


Figure 11. External Data Memory Write Cycle

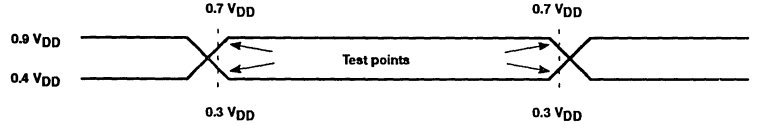


Figure 12. AC Testing Input Waveform

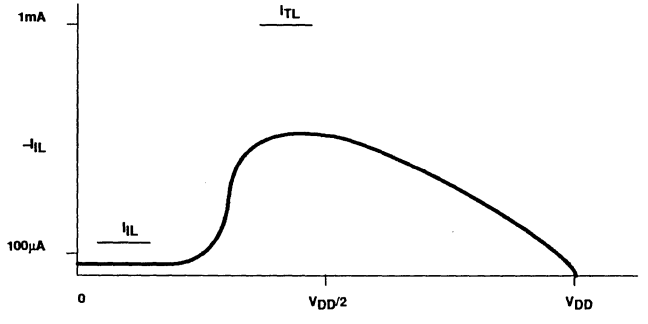
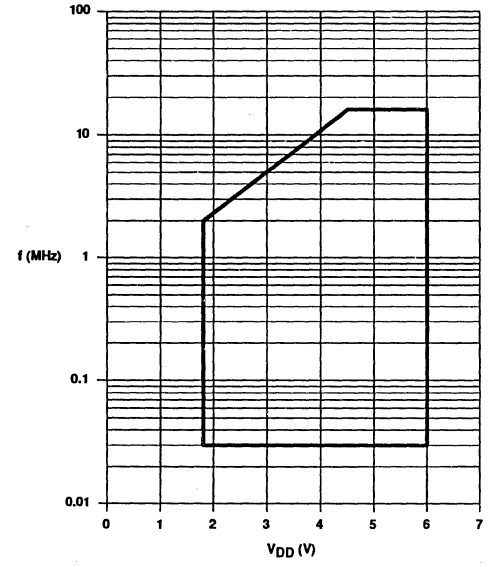


Figure 13. Input Current



NOTE: Below 32kHz, clock has to be supplied externally.

Figure 14. Frequency Operating Range

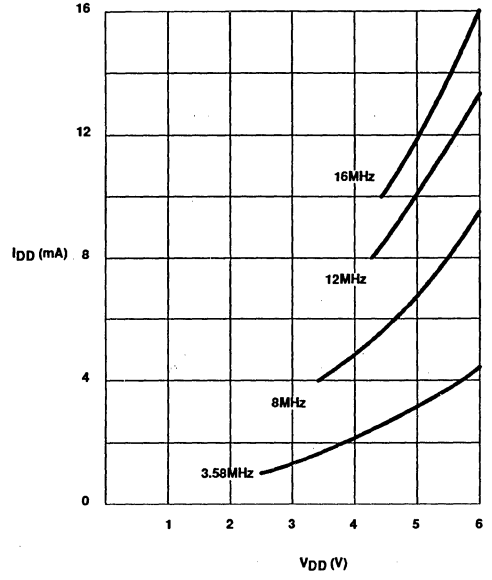


Figure 15. Typical Operating Current as a Function of Frequency and V_{DD}, T_{amb} = 25°C

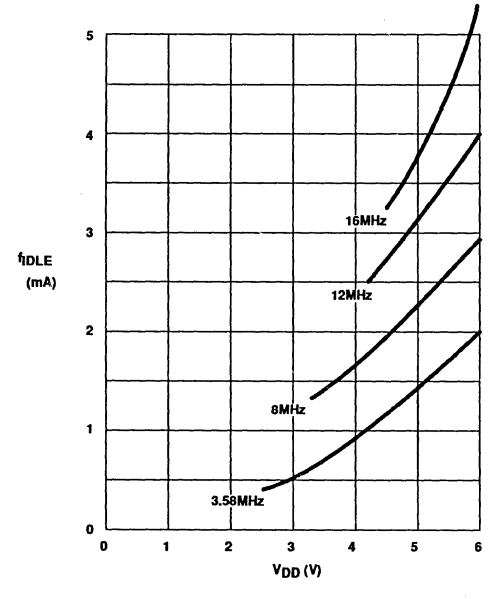


Figure 16. Typical Idle Current as a Function of Frequency and V_{DD}, T_{amb} = 25°C

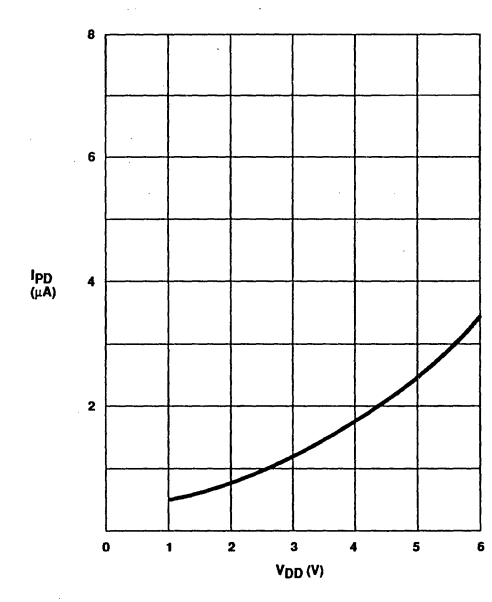


Figure 17. Typical Power-Down Current Vs. Frequency and V_{DD}, T_{amb} = 25°C

Low voltage/low power single-chip 8-bit microcontroller with I^2C

80CL410/83CL410

PIGGYBACK SPECIFICATION

The differences between the masked version and the piggyback are described herein.

General Description

The P85CL000HFZ is a piggy-back version with 256 bytes of RAM used for emulation of the P83CL410 microcontroller. The P85CL000HFZ is manufactured in an advanced CMOS technology. The instruction set of the P85CL000HFZ is based on that of the 8051. The device has low power consumption and a wide supply voltage range. The P85CL000HFZ has two software selectable modes of reduced activity for further power reduction: Idle and Power-down. For timing and AC/DC characteristics, please refer to the P83CL410 specifications.

Features

- Full static 80C51 CPU
- 8-bit CPU, RAM, I/O in a single 40-lead DIP
- Socket for up to 16k external EPROM
- 256 bytes RAM, expandable externally to 64K bytes
- Four 8-bit ports, 32 I/O lines
- Two 16-bit timer/event counters
- External memory expandable up to 128K, external ROM up to 64K and/or RAM up to 64K
- Thirteen source, thirteen vector interrupt structure with two priority levels
- Full duplex serial port (UART)
- I^2C-bus interface for serial transfer on two lines
- Enhanced architecture with:
  - non-page oriented instructions
  - direct addressing
  - four eight byte RAM register banks
  - stack depth up to 128 bytes
  - multiply, divide, subtract and compare instructions
- STOP and IDLE instructions
- Wake-up via external interrupts at port 1
- Single supply voltage of 1.8V to 6.0V
- On-chip oscillator (option: oscillator 4)
- Very low current consumption
- Operating temperature range: –40 to +85°C

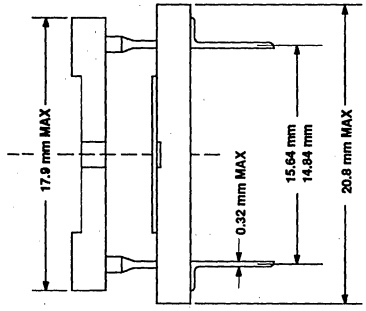
STANDARD PIGGYBACK

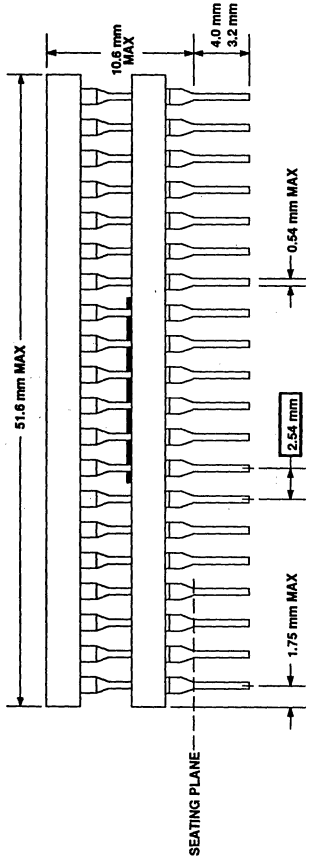
Types: P85CL000HFZ

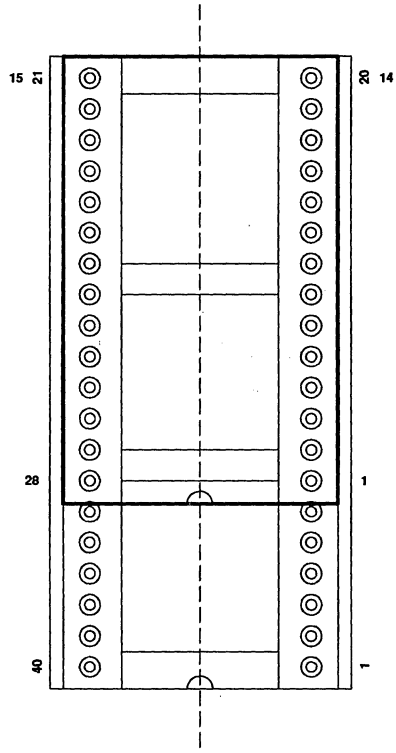
Emulation for: P83CL410, P80CL51

List of differences between masked microcontroller and corresponding piggyback:

PARAMETER	MASKED CONTROLLER	PIGGYBACK
RAM size	128	256
ROM size	4k	EPROM size dependent (max 16k)
Port option	1, 2, 3	1
Oscillator option	32kHz, Osc, 2, 3, 4, RC	Osc. 4
Mech. dimensions	Standard Dual In-Line, Small Outline	See Figure 18
Current cons.	I_{DD}	I_{DD} (OSC. 4) + I_{EPROM}
Voltage range	full	full, limited by EPROM
ESD	specification	not tested (different package)







EPROM

64k	8k	Bytes
128k	16k	Bytes

Figure 18. Piggyback Dimensions
40-Lead Dual In-Line Piggyback: P85CL000HFZ

8XC451 OVERVIEW

The 80C451, the 83C451, and the 87C451 (hereafter referred to collectively as the 8XC451) are I/O expanded versions of the 80C51. Three I/O ports have been added to the basic 80C51 architecture for a total of 7 on-chip I/O ports. The LCC version has a total of 68 pins. The DIP version has 64 pins. Port 6 has 4 control lines to facilitate high-speed asynchronous I/O functions.

The 83C451/87C451 includes a 4k × 8 ROM/EPROM, a 128 × 8 RAM, 56 (LCC) or 52 (DIP) I/O lines, two 16-bit timer/counters, a five source, two priority, level nested interrupt structure, a serial I/O port for either full duplex UART, I/O expansion, or multiprocessor communications, and an on-chip oscillator and clock circuits. The 80C451 includes all of the 83C451 features except the on-board 4k × 8 ROM.

The 8XC451 has two software selectable modes of reduced activity for further power reduction: idle mode and power-down mode. Idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. Power-down mode freezes the oscillator, causing all other chip functions to be inoperative while maintaining the RAM contents.

The 8XC451 features include:

- 80C51 based architecture
- 68-pin LCC and 64-pin DIP packages
- Seven 8-bit I/O ports (LCC version)
- Six 8-bit ports and one 4-bit port (DIP version)
- 4k × 8 ROM or EPROM
- 128 × 8 RAM
- Two 16-bit counter/timers
- Two external interrupts
- External memory addressing capability
  - 64k ROM and 64k RAM
- Low power consumption
  - Idle mode
  - Power-down mode

Differences From the 80C51

Special Function Registers

The SFRs are identical to those of the standard 80C51 with the exception of four registers that have been added to allow control of the three additional I/O ports P4, P5, and P6. The additional registers are P4, P5, P6, and CSR. Registers P4, P5, and P6 function as port latches for ports 4, 5, and 6, respectively. These registers operate identically to those for ports 0 through 3 of the 80C51.

The Control Status Register (CSR) is used to control the mode of operation of port 6 and indicates the current status of port 6. All control status register bits can be read and written by the CPU except bits 0 and 1, which are read only. A Reset writes ones to bits 2-7 and zeros to bits 0 and 1. See Table 1 for the specific function of each bit in the Control Status register.

I/O Port Structure

The 8XC451 has a total of seven parallel I/O ports. The first four ports, P0 through P3, are identical in function to those present on the 80C51 family. The added ports 4 and 5 are identical in function to port 1; that is, they are standard quasi-bidirectional ports with no alternate functions and the standard output drive characteristics. Note that on the 68-pin LCC packages, port 4 is an 8-bit port, while on the 64-pin DIP packages, only the lower four bits of port 4 are available. Port 6 is a specialized 8-bit bidirectional I/O port with internal pullups. This special port can sink/source three LS TTL inputs and drive CMOS inputs without external pullups. The flexibility of this port facilitates high-speed parallel data communications. Port 6 operating modes are controlled by the port 6 Control Status Register (CSR). Port 6 and the CSR are addressed at the Special Function Addresses shown in Table 2. Port 6 can be used as a standard I/O port, or in strobed modes of operation in conjunction with the four port 6 control lines listed below:

ODS	Output data strobe (active low)
IDS	Input data strobe (active low)
BFLAG	Bidirectional I/O pin. Can be programmed to output the Input Buffer Full flag (IBF), input an active low Port Enable (PE) signal, or output a high or low logic level.
AFLAG	Bidirectional I/O pin. Can be programmed to output the Output Buffer Full (OBF) flag, input a register select signal (SEL), or output a high or low logic level.

Port 6 can be used in a number of different ways to facilitate data communication. It can be used as a processor bus interface, as a standard quasi-bidirectional I/O port, or as a parallel printer port (either polled or interrupt driven).

Processor Bus Interface

Port 6 allows the use of an 8XC451 as an element on a microprocessor type bus. The host processor could be a general purpose MPU or the data bus of a microcontroller like the 8XC451 itself. Setting up the 8XC451 as a processor bus interface allows single or multiple microcontrollers to be used on a bus as flexible peripheral processing elements. Applications can include: keyboard scanners, serial I/O controllers, servo controllers, etc.

On reset, port 6 is programmed correctly (that is, Special Function registers CSR and P6) for use as a bus interface. This prevents the interface from disrupting data on the bus of a host processor during power-up.

Standard Quasi-bidirectional I/O Port

To use port 6 as a common I/O port, all of the control pins should be tied to ground. On hardware reset, bits 2-7 of the CSR are set to one. With the control pins grounded, the port's operation and electrical characteristics will be identical to port 1 on the 80C51. No further software initialization is required.

Parallel Printer Port

The 8XC451 has the capacity to permit all of the intelligent features of a common printer to be handled by a single chip. The features of port 6 allow a parallel port to be designed with only line driving and receiving chips required as additional hardware. The onboard UART allows RS232 interfacing with only level shifting chips added. The 8-bit parallel ports 0 to 6 are ample to drive onboard control functions, even when ports are used for external memory access, interrupts, and other functions. The RAM addressing ability of ports 0 to 2 can be used to address up to 64k bytes of a hardware buffer/spooler.

In addition, either end of a parallel interface can be implemented using port 6, and the interfaces can be interrupt driven or polled in either case. For more detailed information on port 6 usage, refer to the application notes contained in Section 4, entitled "80C451 Operation of Port 6" and "256k Centronics Printer Buffer Using the SC87C451 Microcontroller."

Table 1. Control Status Register (CSR)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
MB1	**MB0**	**MA1**	**MA0**	**OBFC**	**IDSM**	**OBF**	**IBF**
BFLAG Mode Select		AFLAG Mode Select		Output Buffer Flag Clear Mode	Input Data Strobe Mode	Output Buffer Full Flag	Input Buffer Full Flag
0/0 = Logic 0 output* 0/1 = Logic 1 output* 1/0 = IBF output 1/1 = PE input (0 = Select) (1 = Disable I/O)		0/0 = Logic 0 output 0/1 = Logic 1 output 1/0 = OBF output* 1/1 = SEL input (0 = Data) (1 = Control/status)		0 = Negative edge of $\overline{ODS}$ 1 = Positive edge of $\overline{ODS}$	0 = Positive edge of $\overline{IDS}$ 1 = Low level of $\overline{IDS}$	0 = Output data buffer empty 1 = Output data buffer full	0 = Input data buffer empty 1 = Input data buffer full

NOTE:
* Output-always mode: MB1 = 0, MA1 = 1, AND MA0 = 0. In this mode, port 6 is always enabled for output. $\overline{ODS}$ only clears the OBF flag.

Table 2. Special Function Register Addresses

REGISTER ADDRESS			BIT ADDRESS							
NAME	**SYMBOL**	**ADDRESS**	**MSB**							**LSB**
Port 4	P4	C0	C7	C6	C5	C4	C3	C2	C1	C0
Port 5	P5	C8	CF	CE	CD	CC	CB	CA	C9	C8
Port 6 data	P6	D8	DF	DE	DD	DC	DB	DA	D9	D8
Port 6 control status	CSR	E8	EF	EE	ED	EC	EB	EA	E9	E8

Table 3. 8X451 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT NAMES AND ADDRESSES MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
			EF	EE	ED	EC	EB	EA	E9	E8	
CSR*#	Port 6 command/status	E8H	MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF	FCH
DPTR	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt priority	B8H	–	–	–	PS	PT1	PX1	PT0	PX0	xxx00000B
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt enable	A8H	EA	–	–	ES	ET1	EX1	ET0	EX0	0xx00000B
P0*	Port 0	80H	87	B6	85	84	83	82	81	80	FFH
P1*	Port 1	90H	97	96	95	94	93	92	91	90	FFH
P2*	Port 2	A0H	A7	A6	A5	A4	A3	A2	A1	A0	FFH
P3*	Port 3	B0H	B7	B6	B5	B4	B3	B2	B1	B0	FFH
P4*#	Port 4	C0H	C7	C6	C5	C4	C3	C2	C1	C0	FFH
P5*#	Port 5	C8H	CF	CE	CD	CC	CB	CA	C9	C8	FFH
P6*#	Port 6	D8H	DF	DE	DD	DC	DB	DA	D9	D8	FFH
PCON	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
SBUF	Serial data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial port control	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer/counter control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
TMOD	Timer/counter mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
TH0	Timer 0 high byte	8CH									00H
TH1	Timer 1 high byte	8DH									00H
TL0	Timer 0 low byte	8AH									00H
TL1	Timer 1 low byte	8BH									00H

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.

CMOS single-chip 8-bit microcontroller — 80C451/83C451/87C451

DESCRIPTION

The Philips 8XC451 is an I/O expanded single-chip microcontroller fabricated with Philips high-density CMOS technology. Philips epitaxial substrate minimizes latch-up sensitivity.

The 8XC451 is a functional extension of the 87C51 microcontroller with three additional I/O ports and four I/O control lines. The LCC version has a total of 68 pins. Four control lines associated with port 6 facilitate high-speed asynchronous I/O functions.

The 8XC451 includes a 4k × 8 ROM (83C451) EPROM (87C451), a 128 × 8 RAM, 56 (LCC) or 52 (DIP) I/O lines, two 16-bit timer/counters, a five source, two priority level, nested interrupt structure, a serial I/O port for either a full duplex UART, I/O expansion, or muti-processor communications, and on-chip oscillator and clock circuits. The 80C451 includes all of the 83C451 features except the on-board 4k × 8 ROM.

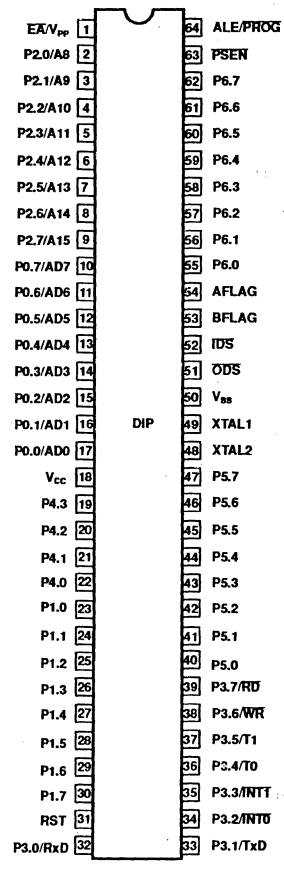
The 87C451 has 4k of EPROM on-chip as program memory and is otherwise identical to the 83C451.

The 8XC451 has two software selectable modes of reduced activity for further power reduction; idle mode and power-down mode. Idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. Power-down mode freezes the oscillator, causing all other chip functions to be inoperative while maintaining the RAM contents.

FEATURES

- 80C51 based architecture
- 68-pin LCC and 64-pin DIP packages:
  - Seven 8-bit I/O ports (LCC version)
  - Six 8-bit ports and one 4-bit port (DIP version)
- Port 6 features:
  - Eight data pins
  - Four control pins
  - Direct MPU bus interface
  - Parallel printer interface
- On the microcontroller:
  - 4k × 8 ROM (83C451)
    4k × 8 EPROM (87C451)
    ROMless version (80C451)
  - 128 × 8 RAM
  - Two 16-bit counter/timers
  - Two external interrupts
- External memory addressing capability
  - 64k ROM and 64k RAM
- Low power consumption:
  - Normal operation: less than 24mA at 5V, 12MHz
  - Idle mode
  - Power-down mode

PIN CONFIGURATIONS

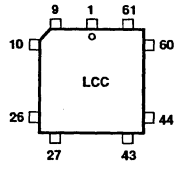


SEE NEXT PAGE FOR LCC PIN FUNCTIONS

PART NUMBER SELECTION

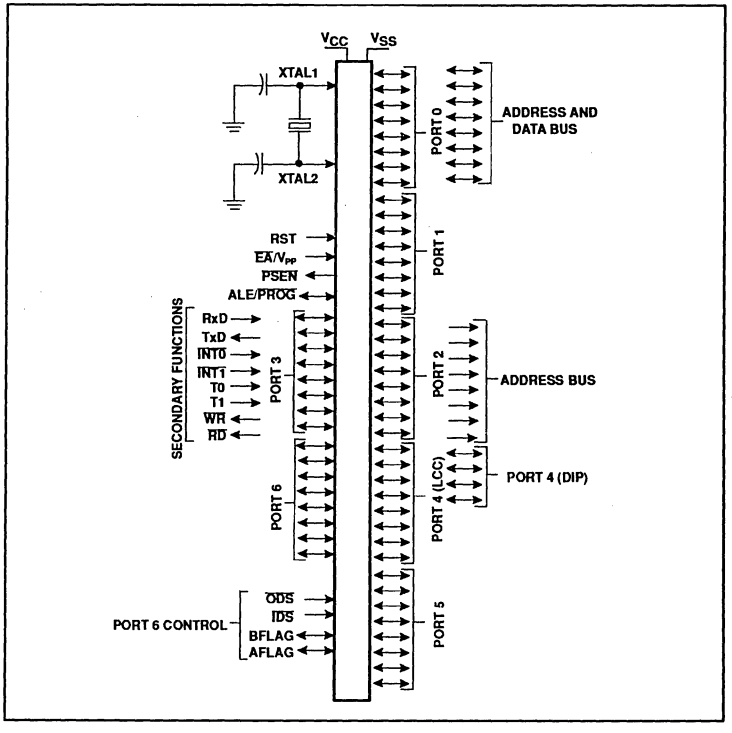
ROMless	ROM	EPROM	TEMPERATURE °C AND PACKAGE	FREQUENCY
SC80C451CCN64	SC83C451CCN64	SC87C451CCN64	0 to +70, plastic DIP	3.5 to 12MHz
SC80C451CGN64	SC83C451CGN64	SC87C451CGN64	0 to +70, plastic DIP	3.5 to 16MHz
SC80C451CCA68	SC83C451CCA68	SC87C451CCA68	0 to +70, plastic LCC	3.5 to 12MHz
SC80C451CGA68	SC83C451CGA68	SC87C451CGA68	0 to +70, plastic LCC	3.5 to 16MHz
SC80C451ACN64	SC83C451ACN64	SC87C451ACN64	–40 to +85, plastic DIP	3.5 to 12MHz
SC80C451AGN64	SC83C451AGN64	SC87C451AGN64	–40 to +85, plastic DIP	3.5 to 16MHz
SC80C451ACA68	SC83C451ACA68	SC87C451ACA68	–40 to +85, plastic LCC	3.5 to 12MHz
SC80C451AGA68	SC83C451AGA68	SC87C451AGA68	–40 to +85, plastic LCC	3.5 to 16MHz
		SC87C451CCIA	0 to +70, ceramic DIP	3.5 to 12MHz
		SC87C451CGIA	0 to +70, ceramic DIP	3.5 to 16MHz
		SC87C451CCL68	0 to +70, ceramic LCC	3.5 to 12MHz
		SC87C451CGL68	0 to +70, ceramic LCC	3.5 to 16MHz
		SC87C451ACIA	–40 to +85, ceramic DIP	3.5 to 12MHz
		SC87C451ACL68	–40 to +85, ceramic LCC	3.5 to 12MHz
		SC87C451AGIA	–40 to +85, ceramic DIP	3.5 to 16MHz
		SC87C451AGL68	–40 to +85, ceramic LCC	3.5 to 16MHz

LCC PIN FUNCTIONS

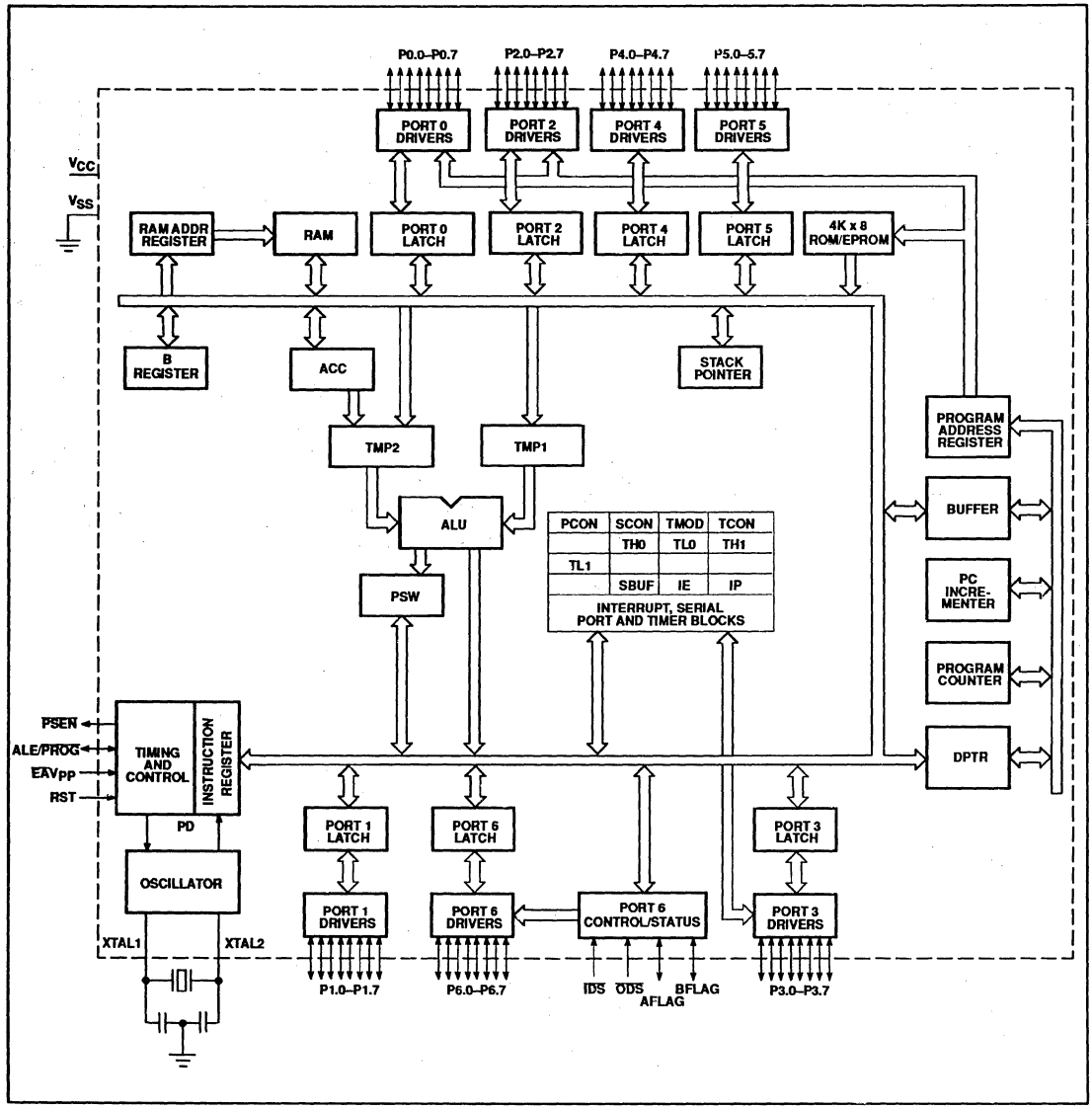


Pin	Function	Pin	Function
1	EA/V_{PP}	35	RST
2	P2.0/A8	36	P3.0/RxD
3	P2.1/A9	37	P3.1/TxD
4	P2.2/A10	38	P3.2/$\overline{INT0}$
5	P2.3/A11	39	P3.3/$\overline{INT1}$
6	P2.4/A12	40	P3.4/T0
7	P2.5/A13	41	P3.5/T1
8	P2.6/A14	42	P3.6/$\overline{WR}$
9	P2.7/A15	43	P3.7/$\overline{RD}$
10	P0.7/AD7	44	P5.0
11	P0.6/AD6	45	P5.1
12	P0.5/AD5	46	P5.2
13	P0.4/AD4	47	P5.3
14	P0.3/AD3	48	P5.4
15	P0.2/AD2	49	P5.5
16	P0.1/AD1	50	P5.6
17	P0.0/AD0	51	P5.7
18	V_{CC}	52	XTAL2
19	P4.7	53	XTAL1
20	P4.6	54	V_{SS}
21	P4.5	55	$\overline{ODS}$
22	P4.4	56	$\overline{IDS}$
23	P4.3	57	BFLAG
24	P4.2	58	AFLAG
25	P4.1	59	P6.0
26	P4.0	60	P6.1
27	P1.0	61	P6.2
28	P1.1	62	P6.3
29	P1.2	63	P6.4
30	P1.3	64	P6.5
31	P1.4	65	P6.6
32	P1.5	66	P6.7
33	P1.6	67	$\overline{PAEN}$
34	P1.7	68	ALE/$\overline{PROG}$

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	50	54	I	**Ground:** 0V reference.
V_{CC}	18	18	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	17-10	17-10	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 is also the multiplexed data and low-order address bus during accesses to external memory. External pull-ups are required during program verification. Port 0 can sink/source eight LS TTL inputs.
P1.0–P1.7	23-30	27-34	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 receives the low-order address bytes during program memory verification. Port 1 can sink/source three LS TTL inputs, and drive CMOS inputs without external pull-ups.
P2.0–P2.7	2-9	2-9	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 emits the high-order address bytes during access to external memory and receives the high-order address bits and control signals during program verification. Port 2 can sink/source three LS TTL inputs, and drive CMOS inputs without external pull-ups.
P3.0–P3.7	32-39	36-43	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 can sink/source three LS TTL inputs, and drive CMOS inputs without external pull-ups. Port 3 also serves the special functions listed below:
	32	36	I	**RxD (P3.0):** Serial input port
	33	37	O	**TxD (P3.1):** Serial output port
	34	38	I	**$\overline{INT0}$ (P3.2):** External interrupt
	35	39	I	**$\overline{INT1}$ (P3.3):** External interrupt
	36	40	I	**T0 (P3.4):** Timer 0 external input
	37	41	I	**T1 (P3.5):** Timer 1 external input
	38	42	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	39	43	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
P4.0–P4.3 P4.0–P4.7	22-19	26-19	I/O I/O	**Port 4:** Port 4 is a 4/8-bit (DIP/LCC) bidirectional I/O port with internal pull-ups. Port 4 can sink/source three LS TTL inputs and drive CMOS inputs without external pull-ups.
P5.0–P5.7	40-47	44-51	I/O	**Port 5:** Port 5 is a 4/8-bit (DIP/LCC) bidirectional I/O port with internal pull-ups. Port 5 can sink/source three LS TTL inputs and drive CMOS inputs without external pull-ups.
P6.o–P6.7	55-62	59-66	I/O	**Port 6:** Port 6 is a specialized 8-bit bidirectional I/O port with internal pull-ups. This special port can sink/source three LS TTL inputs and drive CMOS inputs without external pull-ups. Port 6 can be used in a strobed or non-strobed mode of operation. Port 6 works in conjunction with four control pins that serve the functions listed below:
$\overline{ODS}$	51	55	I	**$\overline{ODS}$:** Output data strobe
IDS	52	56	I	**$\overline{IDS}$:** Input data strobe
BFLAG	53	57	I/O	**BFLAG:** Bidirectional I/O pin with internal pull-ups
AFLAG	54	58	I/O	**AFLAG:** Bidirectional I/O pin with internal pull-ups
RST	31	35	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal pull-down resistor permits a power-on reset using only an external capacitor connected to V_{CC}.
ALE/$\overline{PROG}$	64	68	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. ALE is activated at a constant rate of 1/6 the oscillator frequency except during an external data memory access, at which time one ALE is skipped. ALE can sink/source three LS TTL inputs and drive CMOS inputs without external pull-ups. This pin is also the program pulse during EPROM programming.
$\overline{PSEN}$	63	67	O	**Program Store Enable:** The read strobe to external program memory. $\overline{PSEN}$ is activated twice each machine cycle during fetches from external program memory. However, when executing out of external program memory, two activations of $\overline{PSEN}$ are skipped during each access to external program memory. $\overline{PSEN}$ is not activated during fetches from internal program memory. $\overline{PSEN}$ can sink/source eight LS TTL inputs and drive CMOS inputs without an external pull-up. This pin should be tied low during programming.
$\overline{EA}/V_{PP}$	1	1	I	**Instruction Execution Control/Programming Supply Voltage:** When $\overline{EA}$ is held high, the CPU executes out of internal program memory, unless the program counter exceeds 0FFFH. When $\overline{EA}$ is held low, the CPU executes out of external program memory. $\overline{EA}$ must never be allowed to float. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	49	53	I	**Crystal 1:** Input to the inverting oscillator amplifier that forms the oscillator. This input receives the external oscillator when an external oscillator is used.
XTAL2	48	52	O	**Crystal 2:** An output of the inverting amplifier that forms the oscillator. This pin should be floated when an external oscillator is used.

PORTS 4 AND 5

Ports 4 and 5 are bidirectional I/O ports with internal pull-ups. Port 4 is an 8-bit port (LCC version) or a 4-bit port (DIP version). Port 4 and port 5 pins with ones written to them, are pulled high by the internal pull-ups, and in that state can be used as inputs. Port 4 and 5 are addressed at the special function register addresses shown in Table 1.

PORT 6

Port 6 is a special 8-bit bidirectional I/O port with internal pull-ups (see Figure 1). This port can be used as a standard I/O port, or in strobed modes of operation in conjunction with four special control lines: $\overline{ODS}$, $\overline{IDS}$, AFLAG, and BFLAG. Port 6 operating modes are controlled by the port 6 control status register (CSR). Port 6 and the CSR are addressed at the special function register addresses shown in Table 1. The following four control pins are used in conjunction with port 6:

$\overline{ODS}$ – Output data strobe for port 6. $\overline{ODS}$ can be programmed to control the port 6 output drivers and the output buffer full flag (OBF), or to clear only the OBF flag bit in the CSR (output-always mode). $\overline{ODS}$ is active low for output driver control. the OBF flag can be programmed to be cleared on the negative or positive edge of $\overline{ODS}$.

$\overline{IDS}$ – Input data strobe for port 6. $\overline{IDS}$ is used to control the port 6 input latch and input buffer full flag (IBF) bit in the CSR. The input data latch can be programmed to be transparent when $\overline{IDS}$ is low and latched on the positive transition of $\overline{IDS}$, or to latch only on the positive transition of $\overline{IDS}$. Correspondingly, the IBF flag is set on the negative or positive transition of $\overline{IDS}$.

AFLAG – AFLAG is a bidirectional I/O pin which can be programmed to be an output set high or low under program control, or to output the state of the output buffer full flag. AFLAG can also be programmed to be an input which selects whether the contents of the output buffer, or the contents of the port 6 control status register will output on port 6. This feature grants complete port 6 status to external devices.

BFLAG – BFLAG is a bidirectional I/O pin which can be programmed to be an output, set high or low under program control, or to output the state of the input buffer full flag. BFLAG can also be programmed to input an enable signal for port 6. When BFLAG is used as an enable input, port 6 output drivers are in the high-impedance state, and the input latch does not respond to the $\overline{IDS}$ strobe when BFLAG is high. Both features are enabled when BFLAG is low. This feature facilitates the use of the SC8XC451 in bused multiprocessor systems.

CONTROL STATUS REGISTER

The control status register (CSR) establishes the mode of operation for port 6 and indicates the current status of port 6 I/O registers. All control status register bits can be read and written by the CPU, except bits 0 and 1, which are read only. Reset writes ones to bits 2 through 7, and writes zeros to bits 0 and 1 (see Table 2).

CSR.0 Input Buffer Full Flag (IBF) (Read Only) – The IBF bit is set to a logic 1 when port 6 data is loaded into the input buffer under control of $\overline{IDS}$. This can occur on the negative or positive edge of $\overline{IDS}$, as determined by CSR.2 IBF is cleared when the CPU reads the input buffer register.

CSR.1 Output Buffer Full Flag (OBF) (Read Only) – The OBF flag is set to a logic 1 when the CPU writes to the port 6 output data buffer. OBF is cleared by the positive or negative edge of $\overline{ODS}$, as determined by CSR.3.

CSR.2 IDS Mode Select (IDSM) – When CSR.2 = 0, a low-to-high transition on the $\overline{IDS}$ pin sets the IBF flag. The Port 6 input buffer is loaded on the $\overline{IDS}$ positive edge. When CSR.2 = 1, a high-to-low transition on the $\overline{IDS}$ pin sets the IBF flag. Port 6 input buffer is transparent when $\overline{IDS}$ is low, and latched when $\overline{IDS}$ is high.

CSR.3 Output Buffer Full Flag Clear Mode (OBFC) – When CSR.3 = 1, the positive edge of the $\overline{ODS}$ input clears the OBF flag. When CSR.3 = 0, the negative edge of the $\overline{ODS}$ input clears the OBF flag.

CSR.4, CSR.5 AFLAG Mode Select (MA0, MA1) – Bits 4 and 5 select the mode of operation for the AFLAG pin as follows:

MA1	MA0	AFLAG Function
0	0	Logic 0 output
0	1	Logic 1 output
1	0	OBF flag output (CSR.1)
1	1	Select (SEL) input mode

The select (SEL) input mode is used to determine whether the port 6 data register or the control status register is output on port 6. When the select feature is enabled, the AFLAG input controls the source of port 6 output data. A logic 0 on AFLAG input selects the port 6 data register, and a logic 1 on AFLAG input selects the control status register.

CSR.6, CSR.7 BFLAG Mode Select (MB0, MB1) – Bits 6 and 7 select the mode operation as follows:

MB1	MB0	BFLAG Function
0	0	Logic 0 output
0	1	Logic 1 output
1	0	IBF flag output (CSR.0)
1	1	Port enable ($\overline{PE}$)

In the port enable mode, $\overline{IDS}$ and $\overline{ODS}$ inputs are disabled when BFLAG input is high. When the BFLAG input is low, the port is enabled for I/O.

SPECIAL FUNCTION REGISTER ADDRESSES

Special function register addresses for the device are identical to those of the 80C51, except for the additional registers listed in Table 1.

Table 1. Special Function Register Addresses

REGISTER ADDRESS			BIT ADDRESS							
NAME	SYMBOL	AADDRESS	MSB							LSB
Port 4	P4	C0	C7	C6	C5	C4	C3	C2	C1	C0
Port 5	P5	C8	CF	CE	CD	CC	CB	CA	C9	C8
Port 6 data	P6	D8	DF	DE	DD	DC	DB	DA	D9	D8
Port 6 control status	CSR	E8	EF	EE	ED	EC	EB	EA	E9	E8

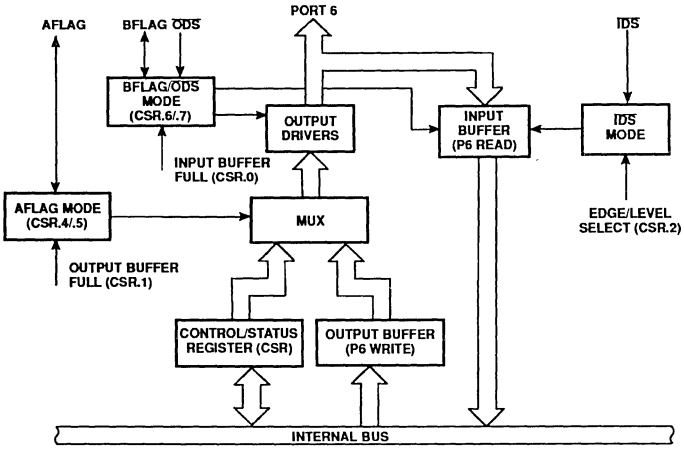


Figure 1. Port 6 Block Diagram

Table 2. Control Status Register (CSR)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MB1	**MB0**	**MA1**	**MA0**	**OBFC**	**IDSM**	**OBF**	**IBF**
BFLAG Mode Select		AFLAG Mode Select		Output Buffer Flag Clear Mode	Input Data Strobe Mode	Output Buffer Flag Full	Input Buffer Flag Full
0/0 = Logic 0 output* 0/1 = Logic 1 output* 1/0 = IBF output 1/1 = PE input (0 = Select) (1 = Disable I/O)		0/0 = Logic 0 output* 0/1 = Logic 1 output* 1/0 = OBF output 1/1 = SEL input (0 = Select) (1 = Control/status)		0 = Negative edge of ODS 1 = Positive edge o ODS	0 = Positive edge of IDS 1 = Low level of IDS	0 = Output data buffer empty 1 = Output data buffer full	0 = Input data buffer empty 1 = Input data buffer full

NOTE:
* Output-always mode: MB1 = 0, MA1 = 1, and MA0 = 0. In this mode, port 6 is always enabled for output. ODS only clears the OBF flag.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Operating temperature under bias	0 to +70 –40 to +85	°C
Storage temperature range	–65 to +150	°C
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.5	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V (80C451, 83C451)
T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (87C451)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			MIN	TYPICAL[1]	MAX	
V_{IL}	Input low voltage; except $\overline{EA}$		–0.5		$0.2V_{CC}$–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$		0		$0.2V_{CC}$–0.3	V
V_{IH}	Input high voltage; except XTAL1, RST		$0.2V_{CC}$+0.9		V_{CC}+0.5	V
V_{IH1}	Input high voltage; XTAL1, RST		$0.7V_{CC}$		V_{CC}+0.5	V
V_{OL}	Output low voltage; ports 1, 2, 3	I_{OL} = 1.6mA[2]			0.45	V
V_{OL1}	Output low voltage; port 0, ALE, $\overline{PSEN}$	I_{OL} = 3.2mA[2]			0.45	V
V_{OH}	Output high voltage; ports 1, 2, 3, 4, 5, 6	I_{OH} = –60µA, I_{OH} = –25µA I_{OH} = –10µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
V_{OH1}	Output high voltage (port 0 in external bus mode, ALE, $\overline{PSEN}$)[3]	I_{OH} = –800µA, I_{OH} = –300µA I_{OH} = –80µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
I_{IL}	Logical 0 input current,; ports 1, 2, 3, 4, 5, 6	V_{IN} = 0.45V			–50	µA
I_{TL}	Logical 1-to-0 transition current; ports 1, 2, 3	See note 4			–650	µA
I_{LI}	Input leakage current; port 0	V_{IN} = V_{IL} or V_{IH}			±10	µA
I_{CC}	Power supply current: Active mode @ 12MHz[5] Idle mode @ 12MHz[5] Power down mode	See note 6		 11.5 1.3 3	 25 4 50	 mA mA µA
R_{RST}	Internal reset pull-down resistor		50		300	kΩ
C_{IO}	Pin capacitance[7] – DIP package – PLCC package				15 10	pF pF

NOTES:

1. Typical ratings are based on a limited number of samples taken from early manufacturing lots and are not guaranteed. The values listed are at room temperature, 5V.
2. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input..
3. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
4. Pins of ports 1, 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
5. I_{CC}MAX at other frequencies is given by:
   Active mode: I_{CC}MAX = 0.94 X FREQ + 13.71
   Idle mode: I_{CC}MAX = 0.14 X FREQ +2.31
   where FREQ is the external oscillator frequency in MHz. I_{CC}MAX is given in mA. See Figure 13.
6. See Figures 14 through 17 for I_{CC} test conditions.
7. C_{IO} applies to ports 1 through 6, AFLAG, BFLAG, XTAL1, XTAL2.

AC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V (80C451, 83C451)[1, 2]
T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (87C451)

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$		Oscillator frequency: **Speed Versions** SC8XC451 B SC8XC451 C SC8XC451 G			 0.5 3.5 3.5	 12 12 16	 MHz MHz MHz
t_{LHLL}	2	ALE pulse width	127		$2t_{CLCL}$–40		ns
t_{AVLL}	2	Address valid to ALE low	28		t_{CLCL}–55		ns
t_{LLAX}	2	Address hold after ALE low	48		t_{CLCL}–35		ns
t_{LLIV}	2	ALE low to valid instruction in		234		$4t_{CLCL}$–100	ns
t_{LLPL}	2	ALE low to $\overline{PSEN}$ low	43		t_{CLCL}–40		ns
t_{PLPH}	2	$\overline{PSEN}$ pulse width	205		$3t_{CLCL}$–45		ns
t_{PLIV}	2	$\overline{PSEN}$ low to valid instruction in		145		$3t_{CLCL}$–105	ns
t_{PXIX}	2	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	2	Input instruction float after $\overline{PSEN}$		59		t_{CLCL}–25	ns
t_{AVIV}	2	Address to valid instruction in		312		$5t_{CLCL}$–105	ns
t_{PLAZ}	2	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	3, 4	$\overline{RD}$ pulse width	400		$6t_{CLCL}$–100		ns
t_{WLWH}	3, 4	$\overline{WR}$ pulse width	400		$6t_{CLCL}$–100		ns
t_{RLDV}	3, 4	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}$–165	ns
t_{RHDX}	3, 4	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	3, 4	Data float after $\overline{RD}$		97		$2t_{CLCL}$–70	ns
t_{LLDV}	3, 4	ALE low to valid data in		517		$8t_{CLCL}$–150	ns
t_{AVDV}	3, 4	Address to valid data in		585		$9t_{CLCL}$–165	ns
t_{LLWL}	3, 4	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}$–50	$3t_{CLCL}$+50	ns
t_{AVWL}	3, 4	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}$–130		ns
t_{QVWX}	3, 4	Data valid to $\overline{WR}$ transition	23		t_{CLCL}–60		ns
t_{WHQX}	3, 4	Data hold after $\overline{WR}$	33		t_{CLCL}–50		ns
t_{RLAZ}	3, 4	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	3, 4	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	t_{CLCL}–40	t_{CLCL}+40	ns
Shift Register							
t_{XLXL}	5	Serial port clock cycle time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	5	Output data setup to clock rising edge	700		$10t_{CLCL}$–133		ns
t_{XHQX}	5	Output data hold after clock rising edge	50		$2t_{CLCL}$–117		ns
t_{XHDX}	5	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	5	Clock rising edge to input data valid		700		$10t_{CLCL}$–133	ns

NOTES: SEE NEXT PAGE

AC ELECTRICAL CHARACTERISTICS (Continued)

T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V (80C451, 83C451)[1, 2]
T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (87C451)

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
Port 6 input (input rise and fall times = 5ns)							
t_{FLFH}	8	$\overline{PE}$ width	270		$3t_{CLCL}$+20		ns
t_{ILIH}	8	$\overline{IDS}$ width	270		$3t_{CLCL}$+20		ns
t_{DVIH}	8	Data setup to $\overline{IDS}$ high or $\overline{PE}$ high	0		0		ns
t_{IHDX}	8	Data hold after $\overline{IDS}$ high or $\overline{PE}$ high	30		30		ns
t_{IVFV}	9	$\overline{IDS}$ to BFLAG (IBF) delay		130		130	ns
Port 6 output							
t_{OLOH}	6	$\overline{ODS}$ width	270		$3t_{CLCL}$+20		ns
t_{FVDV}	7	SEL to data out delay		85		85	ns
t_{OLDV}	6	$\overline{ODS}$ to data out delay		80		80	ns
t_{OHDZ}	6	$\overline{ODS}$ to data float delay		35		35	ns
t_{OVFV}	6	$\overline{ODS}$ to AFLAG (OBF) delay		100		100	ns
t_{FLDV}	6	$\overline{PE}$ to data out delay		120		120	ns
t_{OHFH}	7	$\overline{ODS}$ to AFLAG (SEL) delay	100		100		ns
External Clock							
t_{CHCX}	10	High time	20		20		ns
t_{CLCX}	10	Low time	20		20		ns
t_{CLCH}	10	Rise time		20		20	ns
t_{CHCL}	10	Fall time		20		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

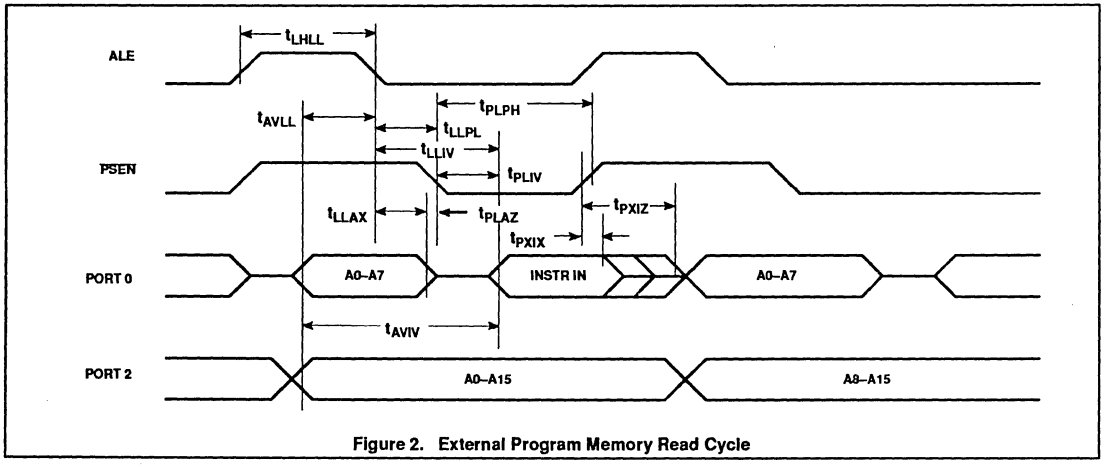


Figure 2. External Program Memory Read Cycle

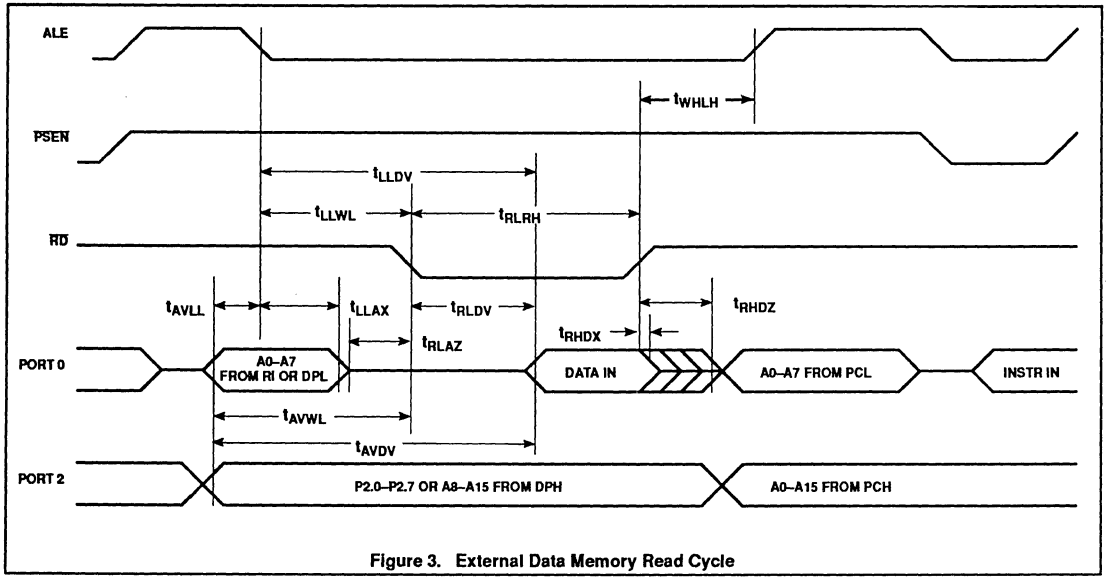


Figure 3. External Data Memory Read Cycle

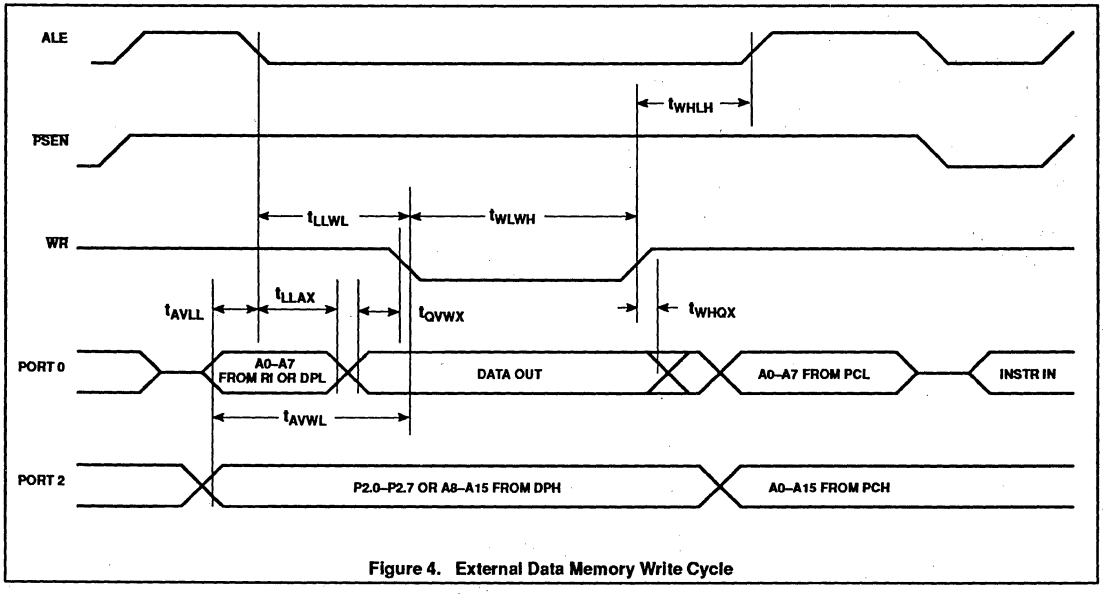


Figure 4. External Data Memory Write Cycle

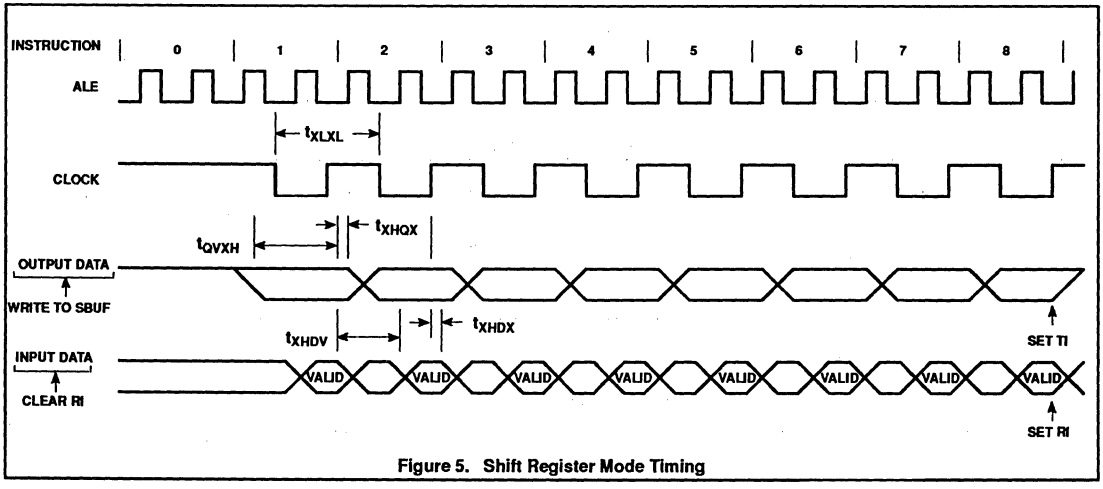


Figure 5. Shift Register Mode Timing

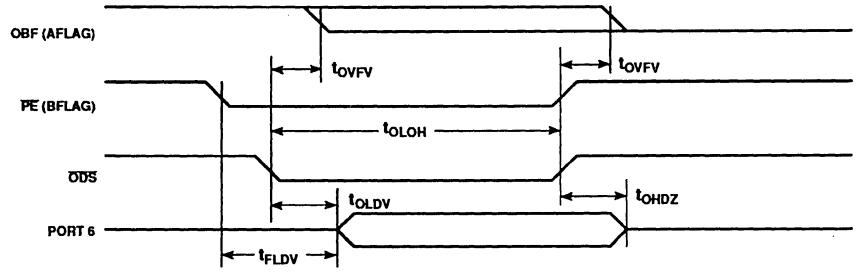


Figure 6. Port 6 Output

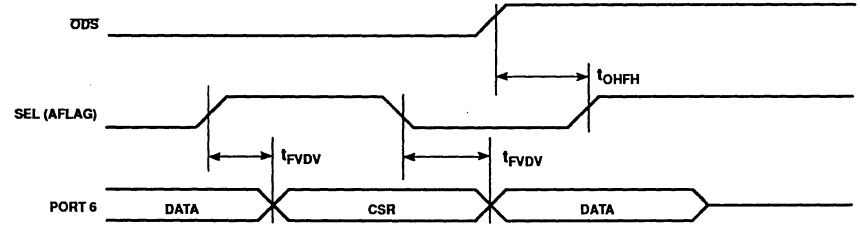


Figure 7. Port 6 Select Mode

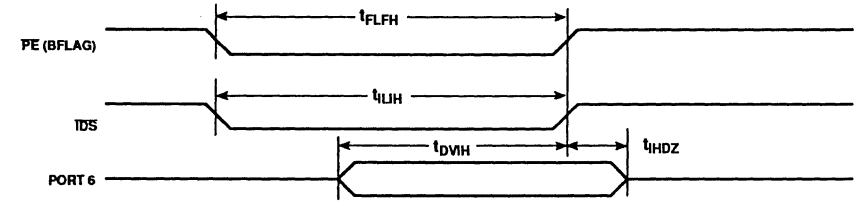


Figure 8. Port 6 Input

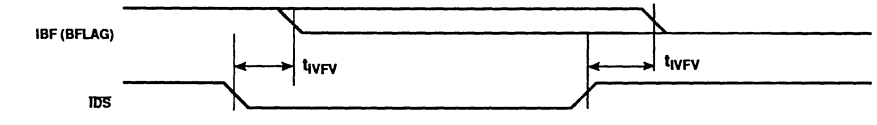


Figure 9. IBF Flag Output

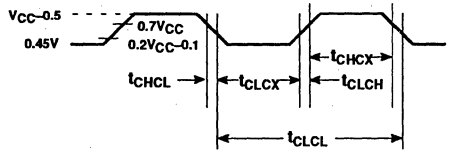
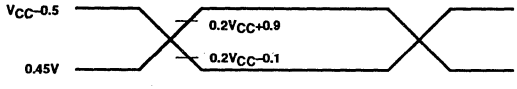
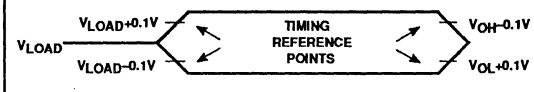


Figure 10. External Clock Drive



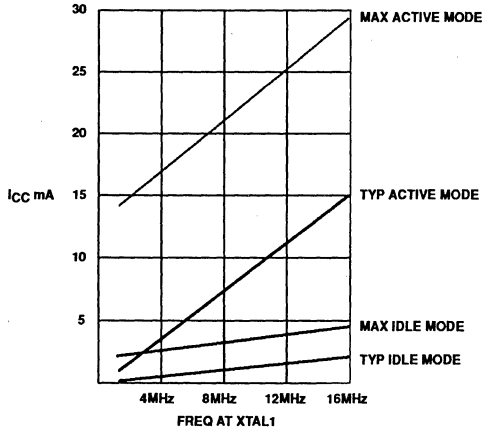
NOTE:
AC inputs during testing are driven at V_{CC} –0.5 for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at V_{IH} min for a logic '1' and V_{IL} for a logic '0'.

Figure 11. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 12. Float Waveform



VALID ONLY WITHIN FREQUENCY SPECIFICATIONS OF THE DEVICE UNDER TEST.

Figure 13. I_{CC} vs. FREQ

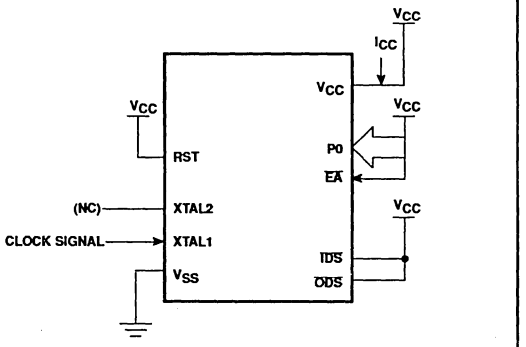


Figure 14. I_{CC} Test Condition, Active Mode
All other pins are disconnected

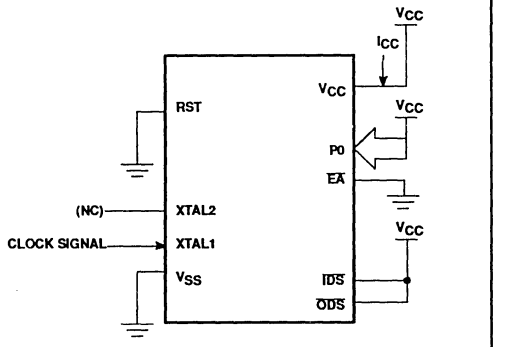


Figure 15. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

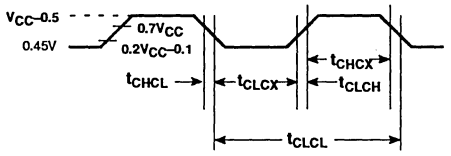


Figure 16. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

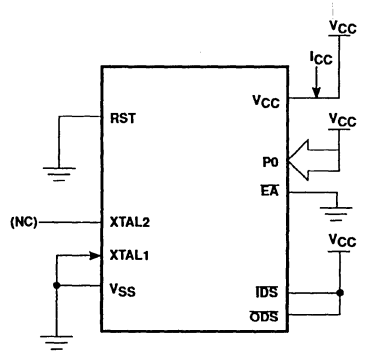


Figure 17. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

EPROM CHARACTERISTICS

The 87C451 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C451 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C451 manufactured by Philips Semiconductors.

Table 3 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 18 and 19. Figure 20 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 18. Note that the 87C451 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 18. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 2 and 3 specified in Table 3 are held at the 'Program Code Data' levels indicated in Table 3. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 19.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 1FH, using the 'Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25 pulse programming sequence using the 'Pgm Lock Bit' levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 1 and 2 as shown in Figure 20. The other pins are held at the 'Verify Code Data' levels indicated in Table 3. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(031H) = 90H indicates 87C451

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 3, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 3. EPROM Progamming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:

1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V ±10% during programming and verification.

* ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

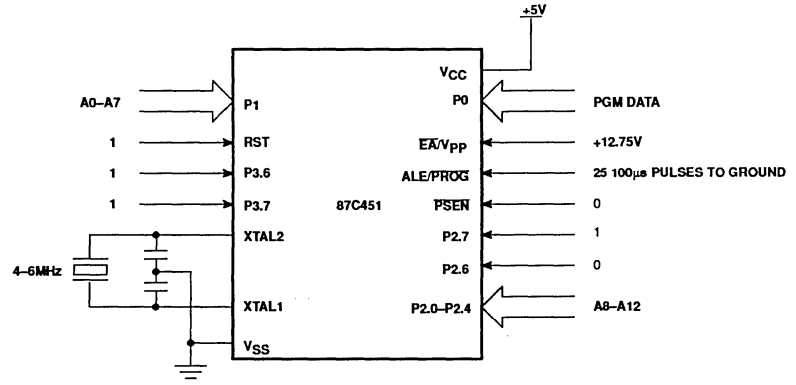


Figure 18. Programming Configuration

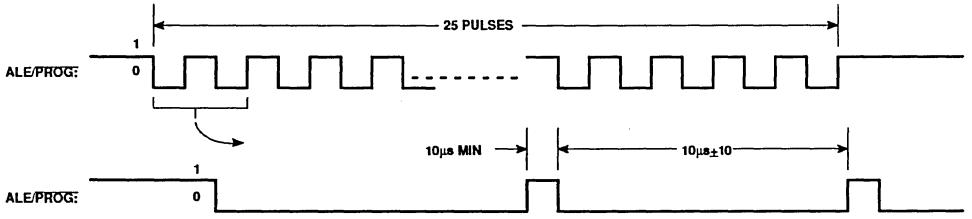


Figure 19. PROG Waveform

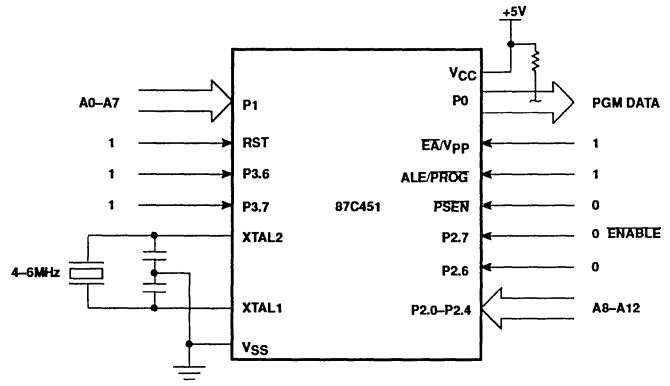


Figure 20. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_{amb} = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 21)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

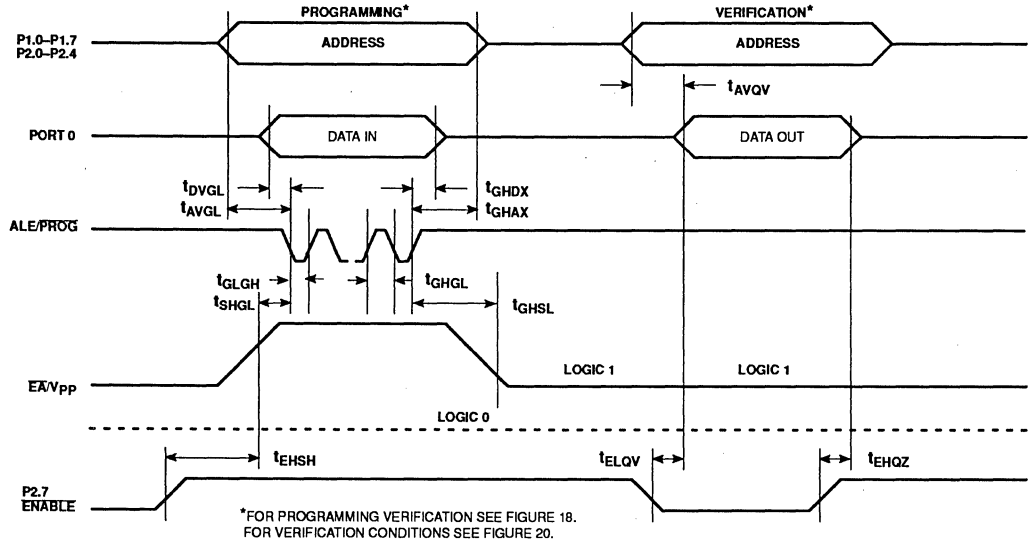


Figure 21. EPROM Programming and Verification

8XC524/8XC528 OVERVIEW

The 8XC524/8XC528 are high-performance single-chip microcontrollers manufactured in an advanced CMOS process and are derivatives of the 80C51 microcontroller family. With the exception of open-drain outputs on two port pins, they are fully compatible with the industry standard 80C51 and include a number of additional enhancements. They are well suited for applications requiring large amounts of on-chip ROM and RAM. Additional special function registers are incorporated to control the on-chip peripherals.

The 8XC524/8XC528 contains a non-volatile 16k (8XC524) or 32k (8XC528) × 8 read-only program memory, a volatile 512 × 8 read/write data memory, four 8-bit I/O ports, two 16-bit timer/counters (identical to those in the 80C51), a 16-bit timer coupled to capture and compare registers (identical to T2 of the 80C52), a multisource two-priority level nested interrupt structure, two serial interfaces (a standard UART and I^2C bus), a watchdog timer with separate oscillator, and a master oscillator. The 8XC524/8XC528 also include ROM code protection and enhanced recovery from power-down mode.

Differences from the 80C51

The 8XC524/8XC528 contains 16 (8XC524) or 32 (8XC528) kbytes of on-chip program memory which can be extended to 64 kbytes with external memories (see Figure 1).

When the $\overline{EA}$ pin is held high, the 8XC524/8XC528 fetches instructions from internal ROM unless the address exceeds 7FFFH. Locations 8000H to FFFFH are fetched from external program memory. When the $\overline{EA}$ pin is held low, all instruction fetches are from external memory.

By setting a mask programmable security bit, the internal ROM contents are protected and cannot be read with the help of test modes or by execution of MOVC instructions from external program memory. The operation of the MOVC instructions in internal and external program memory with the security bit is as follows:

FUNCTION	ACCESS TO INTERNAL PROGRAM MEMORY	ACCESS TO EXTERNAL PROGRAM MEMORY
MOVC in internal program memory	Yes	Yes
MOVC in external program memory	No	Yes

There are no restrictions on the operation of MOVC instructions if the security bit is cleared (logic zero). The state of the EA pin is latched during reset, and its status following reset is ignored. This prevents reading from internal program memory by switching from external to internal mode during the execution of a MOVC instruction.

Data Memory

The internal data memory is divided into four physically separate sections: the lower 128 bytes of RAM, a second 128 bytes of RAM, a 256-byte auxiliary RAM (AUX-RAM), and the 128-byte special function register (SFR) space.

The lower 128 bytes of RAM (addresses 0 to 7FH) are directly and indirectly addressable and correspond to the 128 bytes of RAM in the 80C51. The second 128 bytes of RAM and the special function registers share the same address space but are accessed through different addressing modes.

RAM locations 80H to FFH are only indirectly addressable while the special function registers are only directly addressable. This is the same addressing method used in the 80C52.

The 256-byte AUX-RAM, while physically located on-chip, logically occupies the first 256 bytes of external data memory. As such, it is indirectly addressed in the same way as external data memory using the MOVX instructions in combination with any of the registers R0, R1, or DPTR. Accesses to AUX-RAM locations (0 to FFH) will not affect ports P0, P2, or pins P3.6 or P3.7.

Access to external data memory locations 100H to FFFFH will perform normally. The stack may be located anywhere in the internal data memory by loading the 8-bit stack pointer and has a maximum depth of 256 bytes. The stack may not be located in AUX-RAM. Figure 16 shows the data memory map of the 8XC528 along with a summary of accessing modes.

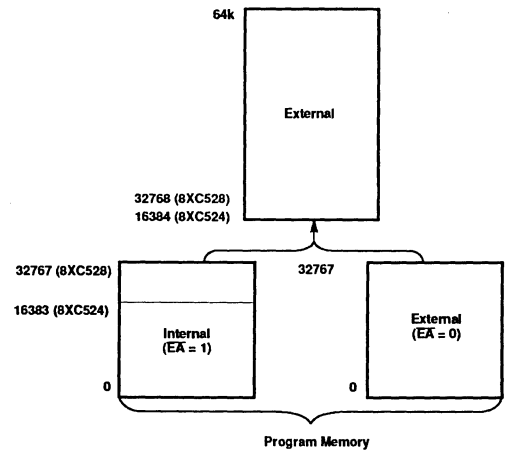


Figure 1. Program Memory

Special Function Registers
The special function registers contain all of the 8XC524/8XC528 registers except the program counter and the four register banks. Most of the 31 special function registers are used to control on-chip peripheral hardware. Other registers include arithmetic registers (ACC, B, PSW), stack pointer (SP), and data pointer registers (DPH, DPL). Thirteen of the SFRs are bit addressable. Table 1 lists the 8XC524/8XC528 special function registers.

The standard 80C52 SFRs are present and function identically in the 8XC524/8XC528. SFRs SCON and SBUF of the 80C51 have been renamed S0CON and S0BUF, respectively.

Watchdog Timer
In addition to timers T0, T1, and T2, the 8XC524/8XC528 also includes a watchdog timer, T3. The purpose of a watchdog timer is to reset the microcontroller within a reasonable time should it enter an erroneous processor state (possibly caused by electrical noise or RFI). When enabled, the watchdog circuitry will generate a system reset if the user program fails to reload the watchdog timer prior to the timer overflowing.

The watchdog timer consists of an 11-bit prescaler and an 8-bit timer, T3, shown in Figure 3. The prescaler is incremented by a dedicated on-chip oscillator with a fixed frequency of 1MHz with a tolerance of +100% and –50%. The 8-bit timer, T3, increments every 2048 cycles of this dedicated oscillator.

When a timer overflow occurs, the 8XC524/8XC528 is reset, and a reset pulse of 16 × 2048 cycles of the dedicated oscillator is generated at the reset pin.

The internal reset signal is not inhibited when the external reset pin is held low by an external circuit. The watchdog timer is controlled by the special function register WDCON. After a reset signal, WDCON will contain the value A5H, which halts the dedicated oscillator and clears both the prescaler and timer T3. Any value stored in WDCON other than A5H will enable the watchdog timer.

Timer T3 can be read on the fly. Timer T3 can only be written if WDCON contains the value 5AH. A successful write operation to T3 will clear the prescaler and WDCON, leaving the watchdog enabled and preventing inadvertent changes of T3.

During a read or write operation to T3, the output of the dedicated oscillator is inhibited to prevent timing problems due to asynchronous increments of T3. To prevent an overflow of the watchdog timer, the user program has to reload the watchdog timer within periods that are shorter than the programmed watchdog interval. This time interval is determined by the 8-bit value loaded into T3.

$$\text{Watchdog interval} = \frac{[256 - (T3)] \times 2048}{\text{dedicated osc freq}}$$

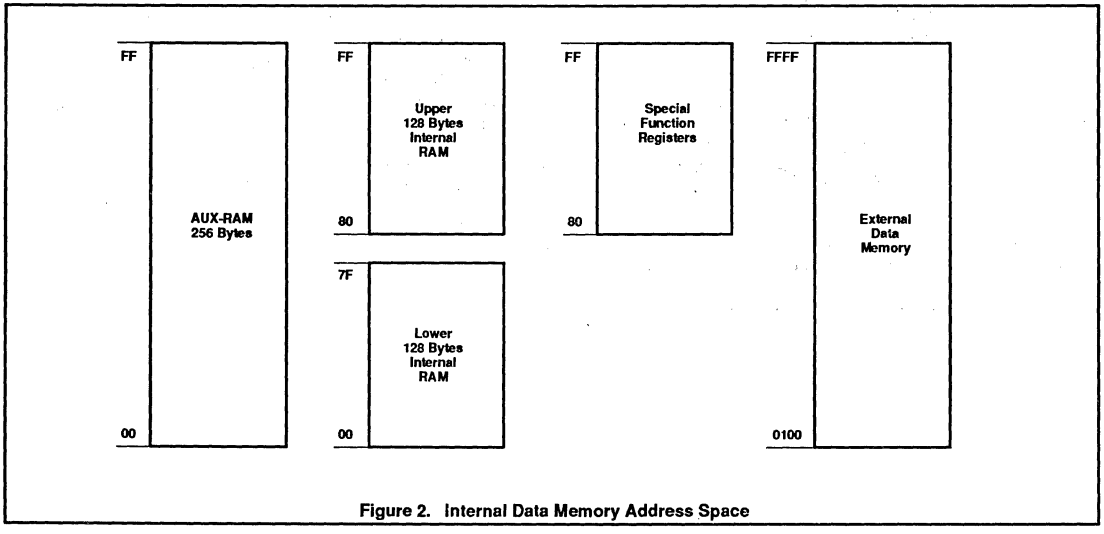


Figure 2. Internal Data Memory Address Space

Table 1. 8XC524/8XC528 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes):										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*#	Interrupt enable	A8H	EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0	00H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*#	Interrupt priority	B8H	–	PS1	PT2	PS0	PT1	PX1	PT0	PX0	x0000000B
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	SDA	SEL	–	–	–	–	T2EX	T2	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	A15	A14	A13	A12	A11	A10	A9	A8	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	RD	WR	T1	T0	INT1	INT0	TxD	RxD	FFH
PCON	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	F1	P	00H
RCAP2H#	Capture high	CBH									00H
RCAP2L#	Capture low	CAH									00H
SBUF	Serial data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial controller	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
S1BIT#	Serial I²C data	D9H/RD	SDI	0	0	0	0	0	0	0	x0000000B
		WR	SD0	X	X	X	X	X	X	X	0xxxxxxxB
S1INT#	Serial I²C interrupt	DAH	INT	X	X	X	X	X	X	X	0xxxxxxxB
			DF	DE	DD	DC	DB	DA	D9	D8	
S1SCS*#	Serial I²C control	D8H/RD	SDI	SCI	CLH	BB	RBF	WBF	STR	ENS	xxxx0000B
		WR	SD0	SC0	CLH	X	X	X	STR	ENS	00xxxx00B
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
			CF	CE	CD	CC	CB	CA	C9	C8	
T2CON*#	Timer 2 control	C8H	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	00H
TH0	Timer high 0	8CH									00H
TH1	Timer high 1	8DH									00H
TH2#	Timer high 2	CDH									00H
TL0	Timer low 0	8AH									C0H
TL1	Timer low 1	8BH									00H
TL2#	Timer low 2	CCH									00H
T3#	Watchdog timer	FFH									00H
TMOD	Timer mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
WDCON#	Watchdog control	A5H									A5H

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.

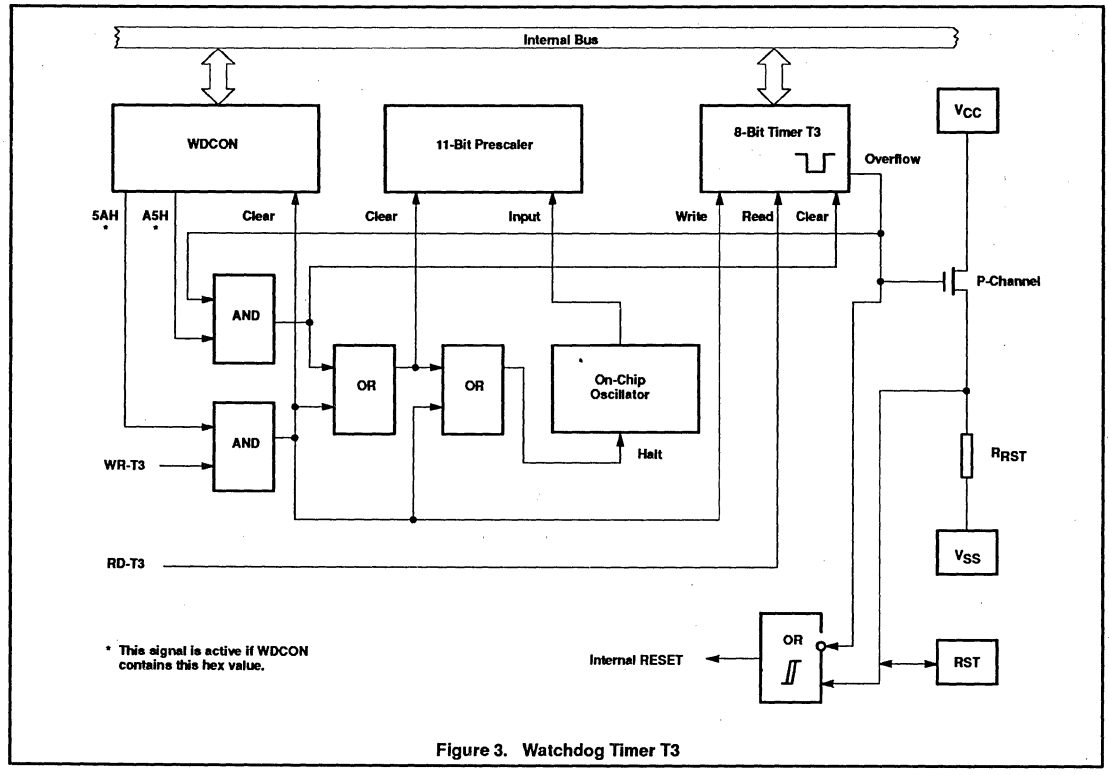


Figure 3. Watchdog Timer T3

I²C Interface

Support of the I²C bus on the 8XC524/8XC528 is provided by a bit-level serial interface. This interface is supported by registers S1INT, S1BIT, and S1SCS in conjunction with pins P1.6/SCL and P1.7/SDA. These latter two pins meet the I²C bus specifications for input and output drive levels and consequently have open drain outputs. All four modes of the I²C bus are supported: master transmitter, master receiver, slave transmitter, and slave receiver. A 100kbps data rate can be achieved in slave mode with a master oscillator frequency of 12MHz. In Master mode with a 12MHz clock a muximum data rate of 70k bps can be achieved.

The I²C interface on the 8XC524/8XC528 performs the following functions:

- Generates an interrupt on reception of a START condition
- Recognizes a STOP condition and indicates busy or free status of bus
- Latches a received serial bit
- Generates a single serial clock pulse on SCL pin
- Performs serial clock synchronization
- Detects bit-level arbitration loss

The three SFRs used for I²C are:

S1INT

7	6	5	4	3	2	1	0
INT	X	X	X	X	X	X	X

S1BIT (READj)

7	6	5	4	3	2	1	0
SDI	0	0	0	0	0	0	0

(WRITE)

7	6	5	4	3	2	1	0
SDO	X	X	X	X	X	X	X

S1SCS (READ)

7	6	5	4	3	2	1	0
SDI	SCI	CLH	BB	RBF	WBF	STR	ENS

(WRITE)

7	6	5	4	3	2	1	0
SDO	SCO	CLH	X	X	X	STR	ENS

Interrupts

The interrupt structure of the 8XC524/8XC528 is the same as that used in the 80C51 but includes two additional interrupt sources: one for the third timer/counter, T2, and one for the I²C interface. The interrupt enable and interrupt priority registers are IE and IP.

IE (A8H)

7	6	5	4	3	2	1	0
EA	ES1	ET2	ES0	ET1	EX1	ET0	EX0

Symbol	Position	Function
EA	IE.7	General enable/disable control 0 – No interrupt is enabled 1 – Any individually enabled interrupt will be accepted
ES1	IE.6	Enable bit-level I²C I/O interrupt
ET2	IE.5	Enable timer 2 interrupt
ES0	IE.4	Enable serial port interrupt
ET1	IE.3	Enable timer 1 interrupt
EX1	IE.2	Enable external 1 interrupt
ET0	IE.1	Enable timer 0 interrupt
EX0	IE.0	Enable external 0 interrupt

IP (B8H)

7	6	5	4	3	2	1	0
–	PS1	PT2	PS0	PT1	PX1	PT0	PX0

Symbol	Position	Function
–	IP.7	Reserved
PS1	IP.6	Bit-level I²C interrupt priority level
PT2	IP.5	Timer 2 interrupt priority level
PS0	IP.4	Serial port interrupt priority level
PT1	IP.3	Timer 1 interrupt priority level
PX1	IP.2	External interrupt 1 priority level
PT0	IP.1	Enable timer 0 interrupt
PX0	IP.0	External interrupt 0 priority level

The interrupt vector locations and the interrupt priorities are:

Source	Priority within Level
Vector Address	Highest
0003H IE0	
002BH TF2+EXF2	
0053H SI (I²C)	
000BH TF0	
0013H IE1	
001BH TF1	
0023H R1+T1	Lowest

Idle Mode

Idle and power-down operation is similar to that used in the 80C51. Idle mode permits the interrupts, serial ports, and timers to function while the CPU operation is halted. During idle mode, the following functions remain active and may generate an interrupt or reset ending the idle mode:

- Timer 0, 1, 2, or 3 (watchdog)
- Standard async UART
- I²C interface
- External interrupts

Idle mode is entered by setting bit PCON.0. Once in the idle mode the CPU status is preserved, and all registers and RAM maintain their data. The status of device pins during idle mode is shown in Table 2. Idle mode is terminated by the activation of any enabled interrupt or by the occurrence of a reset signal (including the watchdog timer overflow).

Power-Down Mode

During power-down mode, the master oscillator is stopped, CPU status is preserved, and all registers and RAM retain their data. The status of device pins during power down is the same as with the idle mode and is shown in Table 2. The power-down mode is terminated by a reset signal (including a watchdog timer overflow), or by the occurrence of either of the two external interrupts.

To terminate power-down mode with the external interrupts, the given interrupt must be programmed to be level-sensitive and must be enabled. The interrupt pin must be held low until the master oscillator has restarted and is stabilized.

Table 2. Status of the External Pins During Idle and Power-Down Modes

MODE	MEMORY	ALE	PSEN	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Port data	Port data	Port data	Port data
Idle	External	1	1	Floating	Port data	Address	Port data
Power-down	Internal	0	0	Port data	Port data	Port data	Port data
Power-down	External	0	0	Floating	Port data	Port data	Port data

CMOS single-chip 8-bit microcontroller — 83C524/87C524

DESCRIPTION

The 8XC524 single-chip 8-bit microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 8XC524 has the same instruction set as the 80C51.

This device provides architectural enhancements that make it applicable in a variety of applications in consumer, telecom and general control systems, especially in those systems which need large ROM and RAM capacity on-chip.

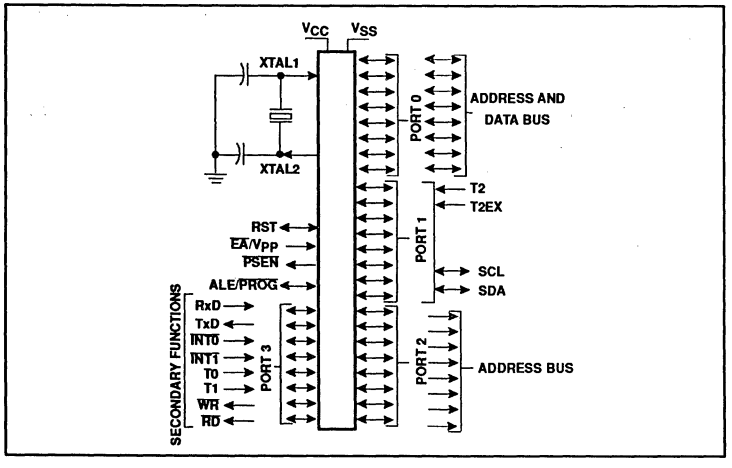
The 8XC524 contains a 16k × 8 ROM (83C524)/EPROM (87C524), a 512 × 8 RAM, four 8-bit I/O ports, two 16-bit timer/event counters (identical to the timers of the 80C51), a 16-bit timer (identical to the timer 2 of the 80C52), a watchdog timer with a separate oscillator, a multi-source, two-priority-level, nested interrupt structure, two serial interfaces (UART and I²C-bus), and on-chip oscillator and timing circuits.

In addition, the 8XC524 has two software selectable modes of power reduction — idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

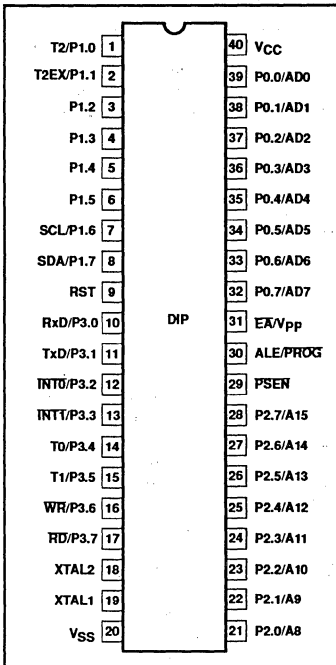
FEATURES

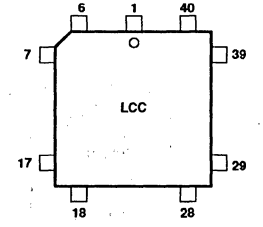
- 80C51 instruction set
  - 16k × 8 ROM (83C524)
  - 16k × 8 EPROM (87C524)
  - 512 × 8 RAM
  - Memory addressing capability 64k ROM and 64k RAM
  - Three 16-bit counter/timers
  - On-chip watchdog timer with oscillator
  - Full duplex UART
  - I²C serial interface
- Power control modes:
  - Idle mode
  - Power-down mode
  - Warm start from power-down
- CMOS and TTL compatible
- Two speed ranges at V_{CC} = 5V ±10%
  - 3.5 to 16MHz
  - 3.5 to 20MHz
- Extended temperature ranges
- OTP package available
- ROM/EPROM code protection

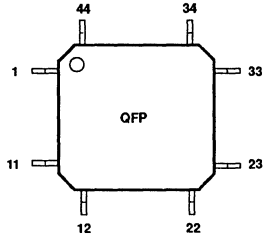
LOGIC SYMBOL



PIN CONFIGURATIONS







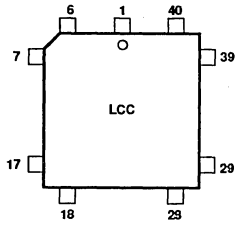
SEE PAGE 303 FOR LCC AND QFP PIN FUNCTIONS.

PART NUMBER SELECTION

ROMless	ROM	EPROM	TEMPERATURE °C AND PACKAGE	FREQUENCY
P80C528FBP N*	P83C524EBP N	P87C524EBP N	0 to +70, plastic DIP	16MHz
		P87C524EBF FA	0 to +70, ceramic DIP with window	16MHz
P80C528FBA A*	P83C524EBA A	P87C524EBA A	0 to +70, plastic PLCC	16MHz
		P87C524EBL KA	0 to +70, ceramic LCC with window	16MHz
P80C528FBB B*	P83C524EBB B	P87C524EBB B	0 to +70, plastic QFP	16MHz
	P83C524GFP N	P87C524GFP N	–40 to +85, plastic DIP	20MHz
		P87C524GFF FA	–40 to +85, ceramic DIP with window	20MHz
	P83C524GFA A	P87C524GFA A	–40 to +85, plastic PLCC	20MHz
		P87C524GFL KA	–40 to +85, ceramic LCC with window	20MHz
	P83C524GFB B	P87C524GFB B	–40 to +85, plastic QFP	20MHz

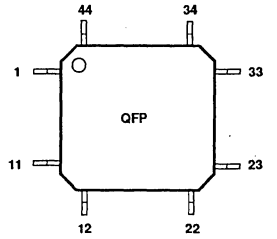
* For full specification, see the 80C528/83C528/87C528 data sheet.

LCC PIN FUNCTIONS



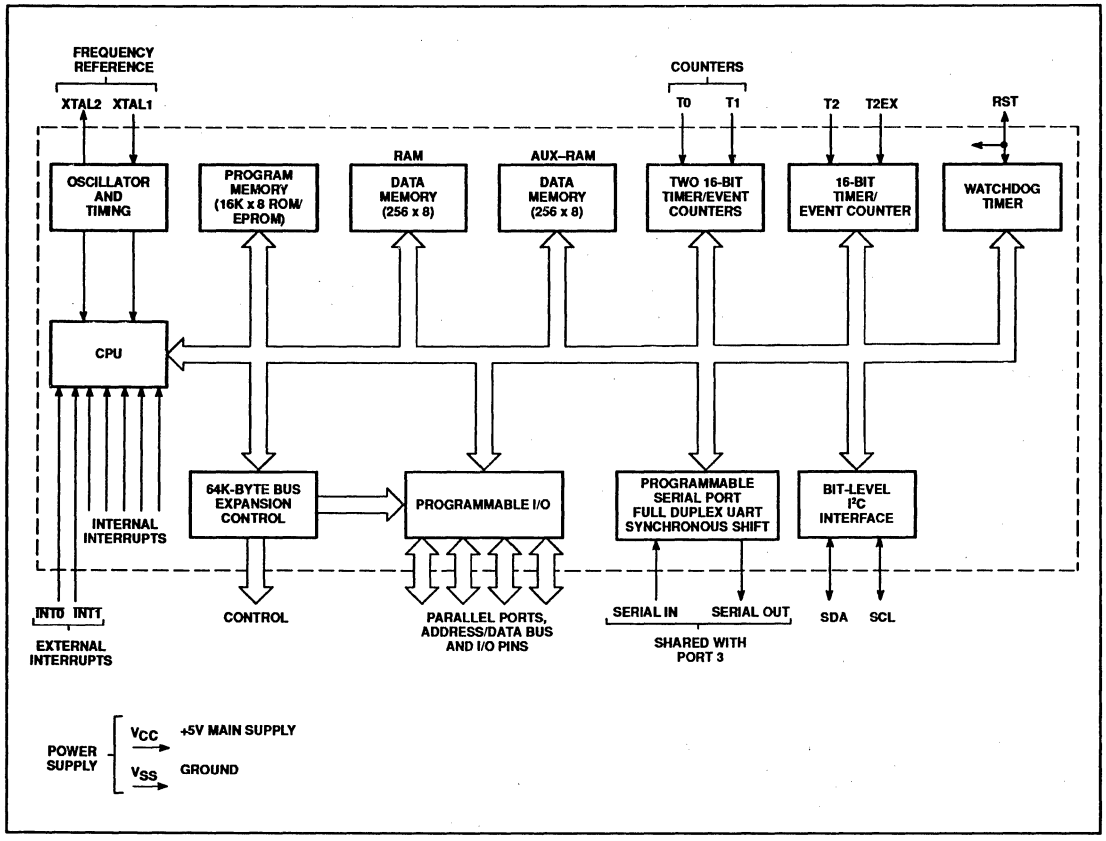
Pin	Function	Pin	Function
1	NC	23	NC
2	P1.0/T2	24	P2.0/A8
3	P1.1/T2EX	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6/SCL	30	P2.6/A14
9	P1.7/SDA	31	P2.7/A15
10	RST	32	$\overline{PSEN}$
11	P3.0/RxD	33	ALE/$\overline{PROG}$
12	NC	34	NC
13	P3.1/TxD	35	$\overline{EA}$/V_{PP}
14	P3.2/$\overline{INT0}$	36	P0.7/AD7
15	P3.3/$\overline{INT1}$	37	P0.6/AD6
16	P3.4/T0	38	P0.5/AD5
17	P3.5/T1	39	P0.4/AD4
18	P3.6/$\overline{WR}$	40	P0.3/AD3
19	P3.7/$\overline{RD}$	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	V_{SS}	44	V_{CC}

QFP PIN FUNCTIONS



Pin	Function	Pin	Function
1	P1.5	23	P2.5/A13
2	P1.6/SCL	24	P2.6/A14
3	P1.7/SDA	25	P2.7/A15
4	RST	26	$\overline{PSEN}$
5	P3.0/RxD	27	ALE/$\overline{PROG}$
6	NC	28	NC
7	P3.1/TxD	29	$\overline{EA}$/V_{PP}
8	P3.2/$\overline{INT0}$	30	P0.7/AD7
9	P3.3/$\overline{INT1}$	31	P0.6/AD6
10	P3.4/T0	32	P0.5/AD5
11	P3.5/T1	33	P0.4/AD4
12	P3.6/$\overline{WR}$	34	P0.3/AD3
13	P3.7/$\overline{RD}$	35	P0.2/AD2
14	XTAL2	36	P0.1/AD1
15	XTAL1	37	P0.0/AD0
16	V_{SS}	38	V_{CC}
17	NC	39	NC
18	P2.0/A8	40	P1.0/T2
19	P2.1/A9	41	P1.1/T2EX
20	P2.2/A10	42	P1.2
21	P2.3/A11	43	P1.3
22	P2.4/A12	44	P1.4

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{SS}	20	22	16	I	**Ground:** 0V reference.
V_{CC}	40	44	38	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–36	37–30	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 also outputs the code bytes during program verification in the P87C524. External pull-ups are required during program verification.
P1.0–P1.7	1–8	2–9	40–44, 1–3	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups, except P1.6 and P1.7 which are open drain. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 1 can sink/source one TTL (4LSTTL) inputs. Port 1 receives the low-order address byte during program memory verification. Port 1 also serves alternate functions for timer 2:
	1	2	40	I	**T2 (P1.0):** Timer/counter 2 external count input.
	2	3	41	I	**T2EX (P1.1):** Timer/counter 2 trigger input.
	7	8	2	I/O	**SCL (P1.6):** I^2C serial port clock line.
	8	9	3	I/O	**SDA (P1.7):** I^2C serial port data line.
P2.0–P2.7	21–28	24–31	18–25	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	5, 7–13	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the SC80C51 family, as listed below:
	10	11	5	I	**RxD (P3.0):** Serial input port
	11	13	7	O	**TxD (P3.1):** Serial output port
	12	14	8	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	9	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	10	I	**T0 (P3.4):** Timer 0 external input
	15	17	11	I	**T1 (P3.5):** Timer 1 external input
	16	18	12	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	13	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	4	I/O	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}. After a watchdog timer overflow, this pin is pulled high while the internal reset signal is active.
ALE/$\overline{PROG}$	30	33	27	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{PSEN}$	29	32	26	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$/V_{PP}	31	35	29	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 7FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 7FFFH. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	19	21	15	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	14	O	**Crystal 2:** Output from the inverting oscillator amplifier.

Table 1. Internal and External Program Memory Access with Security Bit Set

INSTRUCTION	ACCESS TO INTERNAL PROGRAM MEMORY	ACCESS TO EXTERNAL PROGRAM MEMORY
MOVC in internal program memory	YES	YES
MOVC in external program memory	NO	YES

ROM CODE PROTECTION

By setting a mask programmable security bit, the ROM content in the 83C528 is protected, i.e., it cannot be read out by any test mode or by any instruction in the external program memory space. The MOVC instructions are the only ones which have access to program code in the internal or external program memory. The $\overline{EA}$ input is latched during RESET and is 'don't care' after RESET. This implementation prevents reading from internal program code by switching from external program memory to internal program memory during MOVC instruction or an instruction that handles immediate data. Table 1 lists the access to the internal and external program memory by the MOVC instructions when the security bit has been set to logical one. If the security bit has been set to a logical 0 there are no restrictions for the MOVC instructions.

INTERNAL DATA MEMORY

The internal data memory is divided into three physically separated segments: 256 bytes of RAM, 256 bytes of AUX-RAM, and a 128 bytes special function area. These can be addressed each in a different way.

- RAM 0 to 127 can be addressed directly and indirectly as in the 80C51. Address pointers are R0 and R1 of the selected register bank.
- RAM 128 to 255 can only be addressed indirectly as in the 80C51. Address pointers are R0 and R1 of the selected register bank.
- AUX-RAM 0 to 255 is indirectly addressed in the same way as external data memory with the MOVX instructions. Address pointers are R0, R1 of the selected register bank and DPTR. An access to AUX-RAM 0 to 255 will not affect ports P0, P2, P3.6 and P3.7.

An access to external data memory locations higher than 255 will be performed with the MOVX DPTR instructions in the same way as in the 8051 structure, so with P0 and P2 as data/address bus and P3.6 and P3.7 as write and read timing signals. Note that these external data memory cannot be accessed with R0 and R1 as address pointer.

TIMER 2

Timer 2 is functionally equal to the Timer 2 of the 8052AH. Timer 2 is a 16-bit timer/counter. These 16 bits are formed by two special function registers TL2 and TH2. Another pair of special function register RCAP2L and RCAP2H form a 16-bit capture register or a 16-bit reload register. Like Timer 0 and 1, it can operate either as a timer or as an event counter. This is selected by bit C/T2N in the special function register T2CON. It has three operating modes: capture, autoload, and baud rate generator mode which are selected by bits in T2CON.

WATCHDOG TIMER T3

The watchdog timer consists of an 11-bit prescaler and an 8-bit timer formed by special function register T3. The prescaler is incremented by an on-chip oscillator with a fixed frequency of 1MHz. The maximum tolerance on this frequency is –50% and +100%. The 8-bit timer increments every 2048 cycles of the on-chip oscillator. When a timer overflow occurs, the microcontroller is reset and a reset output pulse of 16 × 2048 cycles of the on-chip oscillator is generated at pin RST. The internal RESET signal is not inhibited when the external RST pin is kept low by, for example, an external reset circuit. The RESET signal drives port 1, 2, 3 into the high state and port 0 into the high impedance state.

The watchdog timer is controlled by one special function register WDCON with the direct address location A5H. WDCON can be read and written by software. A value of A5H in WDCON halts the on-chip oscillator and clears both the prescaler and timer T3. After the RESET signal, WDCON contains A5H. Every value other than A5H in WDCON enables the watchdog timer. When the watchdog timer is enabled, it runs independently of the XTAL-clock.

Timer T3 can be read on the fly. Timer T3 can only be written if WDCON contains the value 5AH. A successful write operation to T3 will clear the prescaler and WDCON, leaving the watchdog enabled and preventing inadvertent changes of T3. To prevent an overflow of the watchdog timer, the user program has to reload the watchdog timer within periods that are shorter than the programmed watchdog timer internal. This time interval is determined by an 8-bit value that has to be loaded in register T3 while at the same time the prescaler is cleared by hardware.

Watchdog timer interval =

$$\frac{[256-(T3)] \times 2048}{\text{on-chip oscillator frequency}}$$

BIT-LEVEL I²C INTERFACE

This bit-level serial I/O interface supports the I²C-bus. P1.6/SCL and P1.7/SDA are the serial I/O pins. These two pins meet the I²C specification concerning the input levels and output drive capability. Consequently, these pins have an open drain output configuration. All the four modes of the I²C-bus are supported:

- master transmitter
- master receiver
- slave transmitter
- slave receiver

The advantages of the bit-level I²C hardware compared with a full software I²C implementation are:

- the hardware can generate the SCL pulse
- Testing a single bit (RBF respectively, WBF) is sufficient as a check for error free transmission.

The bit-level I²C hardware operates on serial bit level and performs the following functions:

- filtering the incoming serial data and clock signals
- recognizing the START condition
- generating a serial interrupt request SI after reception of a START condition and the first falling edge of the serial clock
- recognizing the STOP condition
- recognizing a serial clock pulse on the SCL line
- latching a serial bit on the SDA line (SDI)
- stretching the SCL LOW period of the serial clock to suspend the transfer of the next serial data bit
- setting Read Bit Finished (RBF) when the SCL clock pulse has finished and Write Bit

Finished (WBF) if there is no arbitration loss detected (i.e., SDA = 0 while SDO = 1)
- setting a serial clock Low-to-High detected (CLH) flag
- setting a Bus Busy (BB) flag on a START condition and clearing this flag on a STOP condition
- releasing the SCL line and clearing the CLH, RBF and WBF flags to resume transfer of the next serial data bit
- generating an automatic clock if the single bit data register S1BIT is used in master mode.

The following functions must be done in software:
- handling the I²C START interrupts
- converting serial to parallel data when receiving
- converting parallel to serial data when transmitting
- comparing the received slave address with its own
- interpreting the acknowledge information
- guarding the I²C status if RBF or WBF = 0.

Additionally, if acting as master:
- generating START and STOP conditions
- handling bus arbitration
- generating serial clock pulses if S1BIT is not used.

Three SFRs control the bit-level I²C interface: S1INT, S1BIT and S1SCS.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 302.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-up reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-up, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. The power-down mode can be terminated by a RESET in the same way as in the 80C51 or in addition by one of two external interrupts, INT0 or INT1. A termination with an external interrupt does not affect the internal data memory and does not affect the special function registers. This makes it possible to exit power-down without changing the port output levels. To terminate the power-down mode with an external interrupt $\overline{INT0}$ or $\overline{INT1}$ must be switched to level-sensitive and must be enabled. The external interrupt input signal $\overline{INT0}$ and $\overline{INT1}$ must be kept low until the oscillator has restarted and stabilized. An instruction following the instruction that puts the device in the power-down mode will be executed. A reset generated by the watchdog timer terminates the power-down mode in the same way as an external RESET, and only the contents of the on-chip RAM are preserved. The control bits for the reduced power modes are in the special function register PCON.

DESIGN CONSIDERATIONS

At power-on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

When the idle mode is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write when idle is terminated by reset, the instruction following the one that invokes idle should not be one that writes to a port pin or to external memory.

Table 2 shows the state of I/O ports during low current operating modes.

Table 2. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

ROM CODE SUBMISSION

When submitting ROM code for the 83C524, the following must be specified:

1. 16k byte user ROM data
2. 32 byte ROM encryption key
3. ROM security bits.

ADDRESS	CONTENT	BIT(S)	COMMENT
0000H to 7FFFH	DATA	7:0	User ROM Data
8000 to 801FH	KEY	7:0	ROM Encryption Key
8020	SEC	0	ROM Security Bit 1
8020	SEC	1	ROM Security Bit 2
8020	SEC	2	ROM Security Bit 3

Security Bit 1: When programmed, this bit has two effects on masked ROM parts:

1. External MOVC is disabled, and
2. EA# is latched on Reset.

Security Bit 2: When programmed, this bit inhibits Verify User ROM.

Security Bit 3: When programmed, external execution is disabled. Internal data RAM is not accessable.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Operating temperature under bias	0 to +70 or –40 to +85	°C
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ pin to V_{SS}	0 to +13.0	V
Voltage on any other pin to V_{SS}	–0.5 to V_{CC} +0.5	V
Input, output current on any two pins	±10	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.0	W

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V

SYMBOL	PARAMETER	PART TYPE	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{IL}	Input low voltage, except $\overline{EA}$, P1.6/SCL, P1.7/SDA	0 to +70°C –40 to +85°C		–0.5 –0.5	$0.2V_{CC}$–0.1 $0.2V_{CC}$–0.15	V V
V_{IL1}	Input low voltage to $\overline{EA}$	0 to +70°C –40 to +85°C		0 0	$0.2V_{CC}$–0.3 $0.2V_{CC}$–0.35	V V
V_{IL2}	Input low voltage to P1.6/SCL, P1.7SDA[5]			–0.5	1.5	V
V_{IH}	Input high voltage, except XTAL1, RST, P1.6/SCL, P1.7/SDA	0 to +70°C –40 to +85°C		$0.2V_{CC}$+0.9 $0.2V_{CC}$+1	V_{CC}+0.5 V_{CC}+0.5	V V
V_{IH1}	Input high voltage, XTAL1, RST	0 to +70°C –40 to +85°C		$0.7V_{CC}$ $0.7V_{CC}$+0.1	V_{CC}+0.5 V_{CC}+0.5	V V
V_{IH2}	Input high voltage, P1.6/SCL, P1.7/SDA[5]			3.0	6.0	V
V_{OL}	Output low voltage, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA		I_{OL} = 1.6mA[1]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$		I_{OL} = 3.2mA[1]		0.45	V
V_{OL2}	Output low voltage, P1.6/SCL, P1.7/SDA		I_{OL} = 3.0mA[1]		0.4	V
V_{OH}	Output high voltage, ports 1, 2, 3[2]		I_{OH} = –60µA I_{OH} = –25µA I_{OH} = –10µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$		V V V
V_{OH1}	Output high voltage, Port 0 in external bus mode, ALE, $\overline{PSEN}$, RST		I_{OH} = –800µA I_{OH} = –300µA I_{OH} = –80µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$		V V V
I_{IL}	Logical 0 input current, ports 1, 2, 3, except P1.6/SCL, P1.7SDA	0 to +70°C –40 to +85°C	V_{IN} = 0.45V		–50 –75	µA µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA	0 to +70°C –40 to +85°C	See Note 3		–650 –750	µA µA
I_{L1}	Input leakage current, port 0		V_{IN} = V_{IL} or V_{IH}		±10	µA
I_{L2}	Input leakage current, P1.6/SCL, P1.7/SDA		0V < V_I < 6V 0V < V_{CC} < 5.5V		±10	µA
I_{CC}	Power supply current: Active mode @ 16MHz Idle mode @ 16MHz Power down mode	 0 to +70°C –40 to +85°C 0 to +70°C –40 to +85°C	See Note 4		 25 35 5 6 50	 mA mA mA mA µA
R_{RST}	Internal reset pull-down resistor			50	300	kΩ
C_{IO}	Pin capacitance				10	pF

NOTES:

1. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows: 10mA per port pin, port 0 total (all bits) 26mA, ports 1, 2, and total each (all bits) 15mA.
2. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
3. Pins of ports 1, 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
4. See Figures 10 through 13 for I_{CC} test conditions.
5. The input threshold voltage of P1.6 and P1.7 (SI01) meets the I^2C specification, so an input voltage below 1.5V will be recognized as a logic 0 while an input voltage above 3.0V will be recognized as a logic 1.

AC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or −40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V[1,2]

SYMBOL	FIGURE	PARAMETER	16MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	1	Oscillator frequency: **Speed Versions** 8XC524 E 8XC524 G			 3.5 3.5	 16 20	 MHz MHz
t_{LHLL}	1	ALE pulse width	85		$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	8		$t_{CLCL}-55$		ns
t_{LLAX}	1	Address hold after ALE low	28		$t_{CLCL}-35$		ns
t_{LLIV}	1	ALE low to valid instruction in		150		$4t_{CLCL}-100$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	23		$t_{CLCL}-40$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	143		$3t_{CLCL}-45$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		83		$3t_{CLCL}-105$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		38		$t_{CLCL}-25$	ns
t_{AVIV}	1	Address to valid instruction in		208		$5t_{CLCL}-105$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	275		$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	275		$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		148		$5t_{CLCL}-165$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		55		$2t_{CLCL}-70$	ns
t_{LLDV}	2, 3	ALE low to valid data in		350		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		398		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	138	238	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	120		$4t_{CLCL}-130$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	3		$t_{CLCL}-60$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	13		$t_{CLCL}-50$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	23	103	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	6	High time	20		20		ns
t_{CLCX}	6	Low time	20		20		ns
t_{CLCH}	6	Rise time		20		20	ns
t_{CHCL}	6	Fall time		20		20	ns
Shift Register							
t_{XLXL}	4	Serial port clock cycle time	750		$12t_{CLCL}$		ns
t_{QVXH}	4	Output data setup to clock rising edge	492		$10t_{CLCL}-133$		ns
t_{XHQX}	4	Output data hold after clock rising edge	8		$2t_{CLCL}-117$		ns
t_{XHDX}	4	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	4	Clock rising edge to input data valid		492		$10t_{CLCL}-133$	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.

AC ELECTRICAL CHARACTERISTICS – I²C INTERFACE

SYMBOL	PARAMETER	INPUT	OUTPUT	I²C SPECIFICATION
SCL Timing Characteristics				
$t_{HD;STA}$	START condition hold time	≥ 14 t_{CLCL}[1]	Note 2	≥ 4.0μs
t_{LOW}	SCL LOW time	≥ 16 t_{CLCL}	Note 2	≥ 4.7μs
t_{HIGH}	SCL HIGH time	≥ 14 t_{CLCL}[1]	≥ 80 t_{CLCL}[3]	≥ 4.0μs
t_{RC}	SCL rise time	≤ 1μs[4]	Note 5	≤ 1.0μs
t_{FC}	SCL fall time	≤ 0.3μs[4]	≤ 0.3μs [6]	≤ 0.3μs
SDA Timing Characteristics				
$t_{SU;DAT}$	Data set-up time	≥ 250ns	Note 2	≥ 250ns
$t_{HD;DAT}$	Data hold time	≥ 0ns	Note 2	≥ 0ns
$t_{SU;STA}$	Repeated START set-up time	≥ 14 t_{CLCL}[1]	Note 2	≥ 4.7μs
$t_{SU;STO}$	STOP condition set-up time	≥ 14 t_{CLCL}[1]	Note 2	≥ 4.0μs
t_{BUF}	Bus free time	≥ 14 t_{CLCL}[1]	Note 2	≥ 4.7μs
t_{RD}	SDA rise time	≤ 1μs[4]	Note 5	≤ 1.0μs
t_{FD}	SDA fall time	≤ 0.3μs[4]	≤ 0.3μs [6]	≤ 0.3μs

NOTES:
1. At f_{CLK} = 3.5MHz, this evaluates to 14 × 286ns = 4μs, i.e., the bit-level I²C interface can respond to the I²C protocol for f_{CLK} ≥ 3.5MHz.
2. This parameter is determined by the user software, it has to comply with the I²C specification.
3. This value gives the autoclock pulse length which meets the I²C specification for the specified XTAL clock frequency range. Alternatively, the SCL pulse may be timed by software.
4. Spikes on SDA and SCL lines with a duration of less than 4 × f_{CLK} will be filtered out.
5. The rise time is determined by the external bus line capacitance and pull-up resistor, it must be ≤ 1μs.
6. The maximum capacitance on bus lines SDA and SCL is 400pF.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

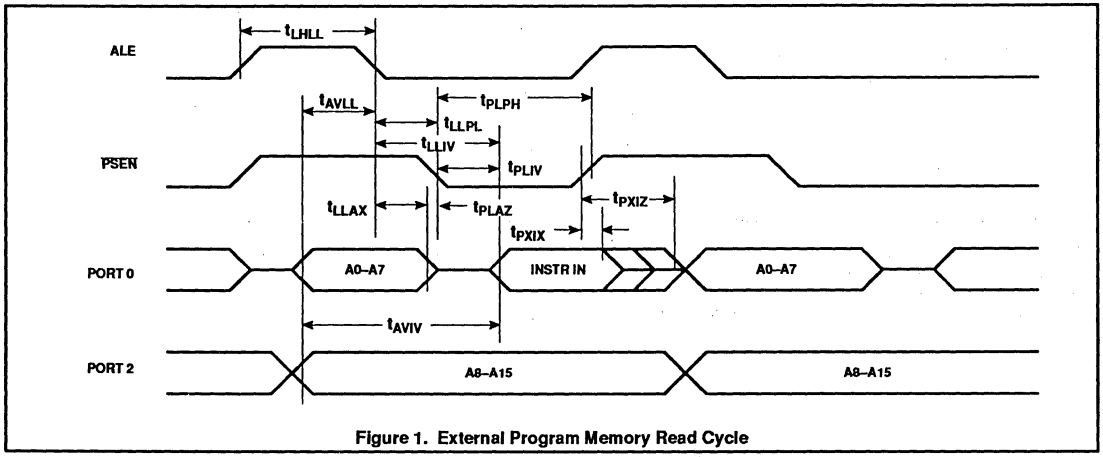


Figure 1. External Program Memory Read Cycle

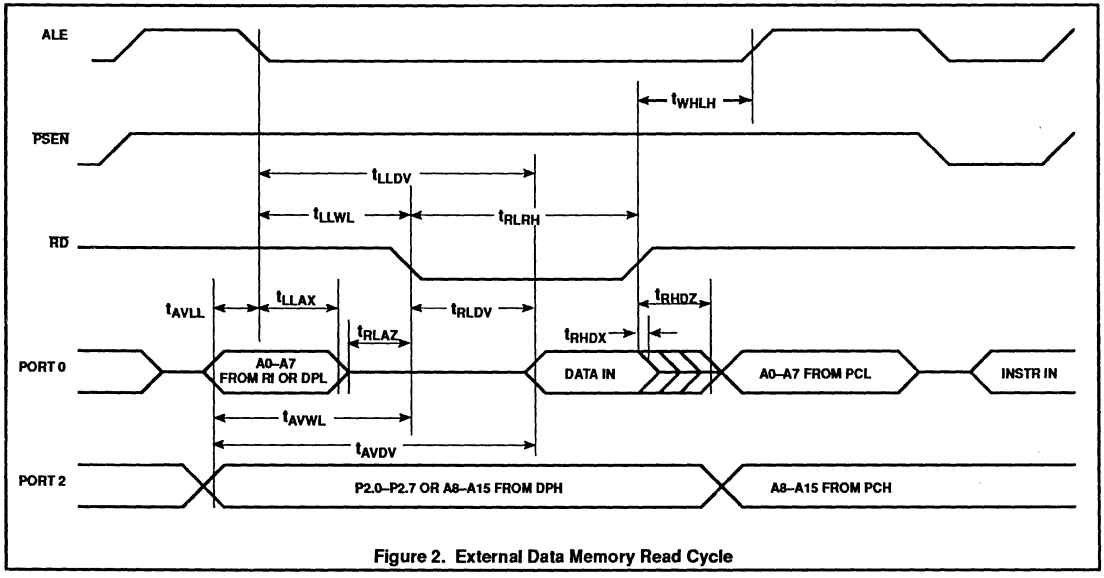


Figure 2. External Data Memory Read Cycle

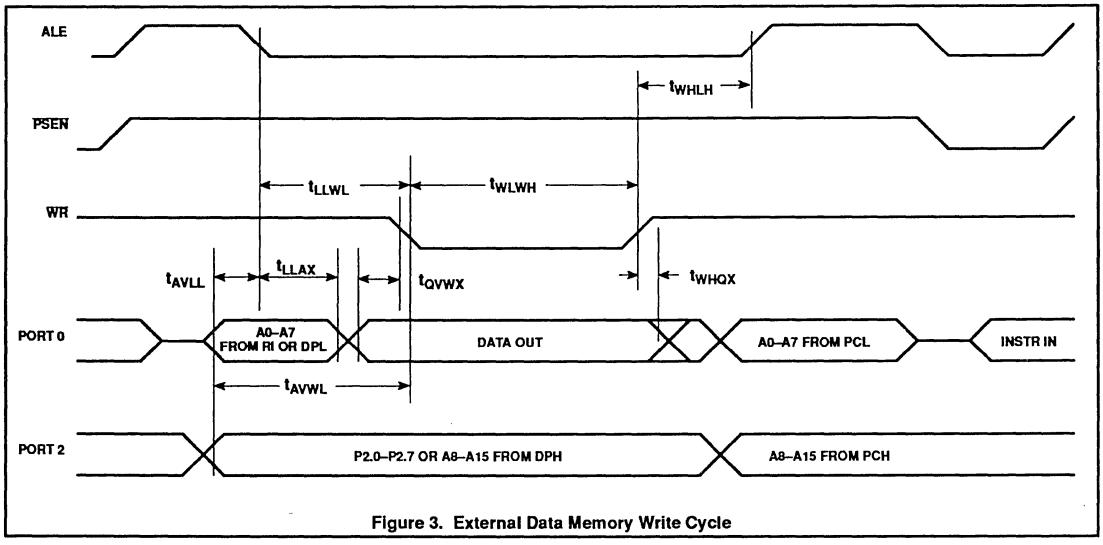


Figure 3. External Data Memory Write Cycle

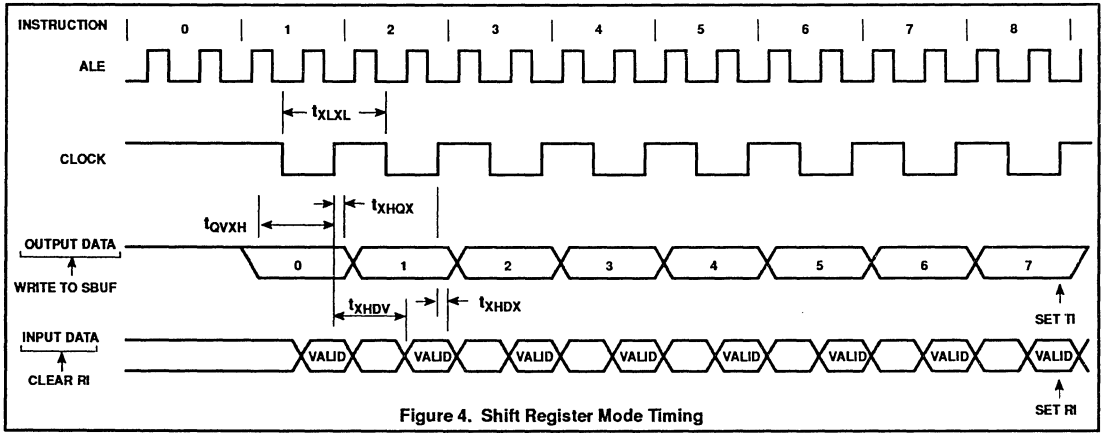


Figure 4. Shift Register Mode Timing

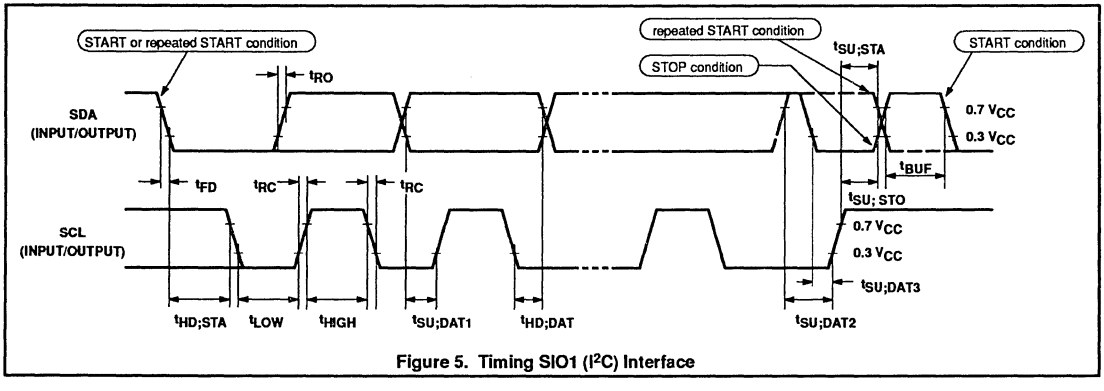


Figure 5. Timing SIO1 (I^2C) Interface

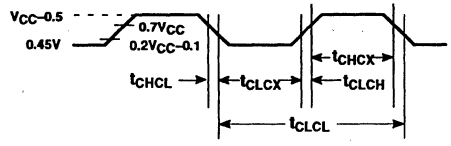
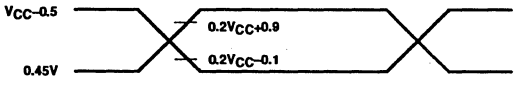
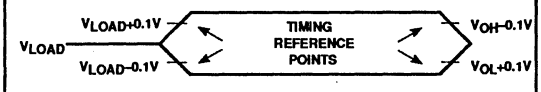


Figure 6. External Clock Drive



NOTE:
AC inputs during testing are driven at V_{CC} –0.5 for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at V_{IH} min for a logic '1' and V_{IL} for a logic '0'.

Figure 7. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 8. Float Waveform

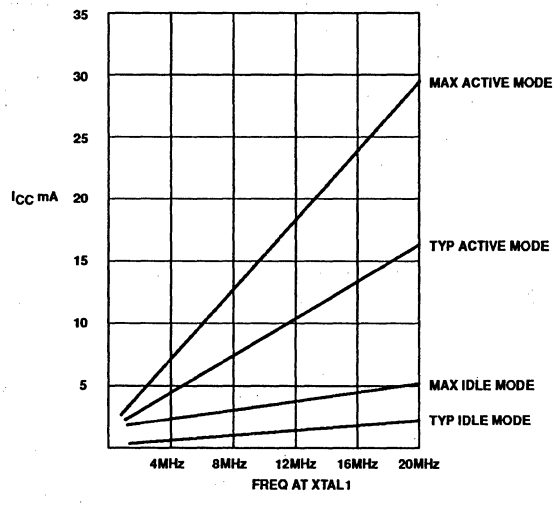


Figure 9. I_{CC} vs. FREQ
Valid only within frequency specifications of the device under test

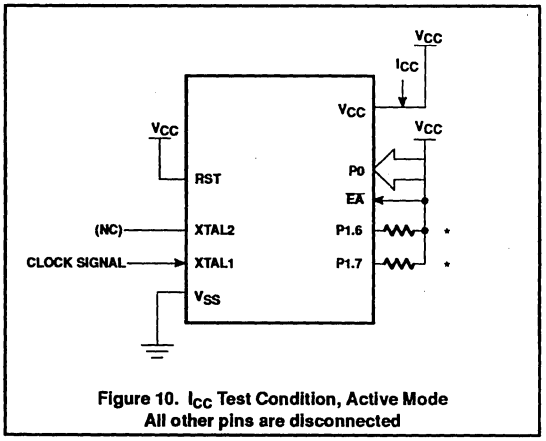


Figure 10. I_{CC} Test Condition, Active Mode
All other pins are disconnected

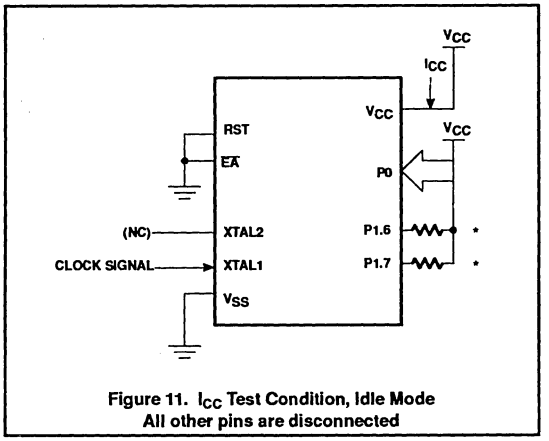


Figure 11. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

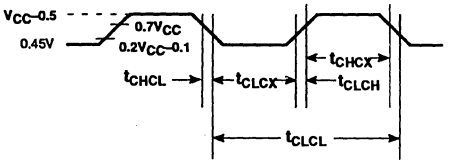


Figure 12. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

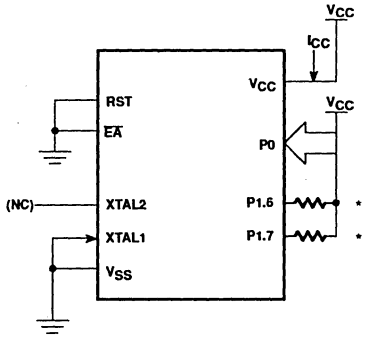


Figure 13. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

NOTE:
* Ports 1.6 and 1.7 should be connected to V_{CC} through resistors of sufficiently high value such that the sink current into these pins does not exceed the I_{OL1} specification.

EPROM CHARACTERISTICS

The 87C524 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C524 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C524 manufactured by Philips.

Table 3 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 14 and 15. Figure 16 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 14. Note that the 87C524 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1, 2 and 3, as shown in Figure 14. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 2 and 3 specified in Table 3 are held at the 'Program Code Data' levels indicated in Table 3. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 15.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 3FH, using the 'Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25 pulse programming sequence using the 'Pgm Lock Bit' levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 1, 2 and 3 as shown in Figure 16. The other pins are held at the 'Verify Code Data' levels indicated in Table 3. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Program Lock Bits

The 87C524 has 3 programmable lock bits that will provide different levels of protection for the on-chip code and data. (See Table 4.)

Erasing the EPROM also erases the encryption array and the program lock bits, returning the part to full functionality.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(031H) = 9DH indicates 87C524

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 3, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

™Trademark phrase of Intel Corporation.

Table 3. EPROM Progamming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0
Pgm lock bit 3	1	0	0*	V_{PP}	0	1	0	1

NOTES:
1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.

* ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

Table 4.

PROGRAM LOCK BITS[1,2]				
	LB1	LB2	LB3	PROTECTION DESCRIPTION
1	U	U	U	No Program Lock features enabled. (Code verify will still be encrypted by the Encryption Array if programmed.)
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on Reset, and further programming of the EPROM is disabled.
3	P	P	U	Same as 2, also verify is disabled.
4	P	P	P	Same as 3, external execution is disabled. Internal data RAM is not accessable.

NOTES:
1. P – programmed. U – unprogrammed.
2. Any other combination of the lock bits is not defined.

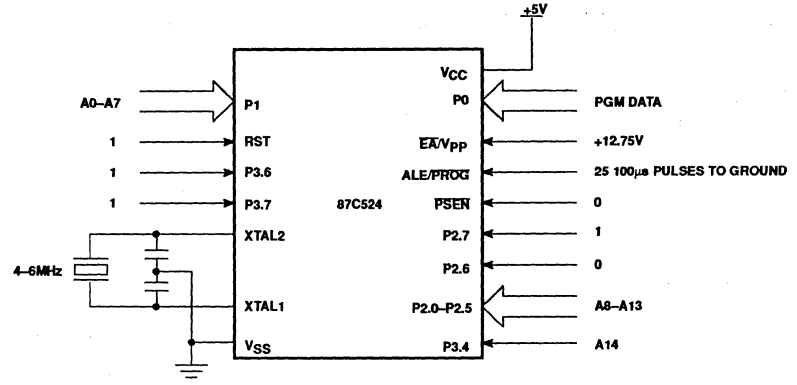


Figure 14. Programming Configuration

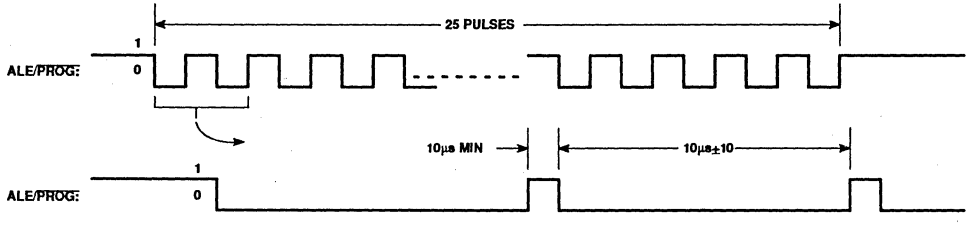


Figure 15. PROG Waveform

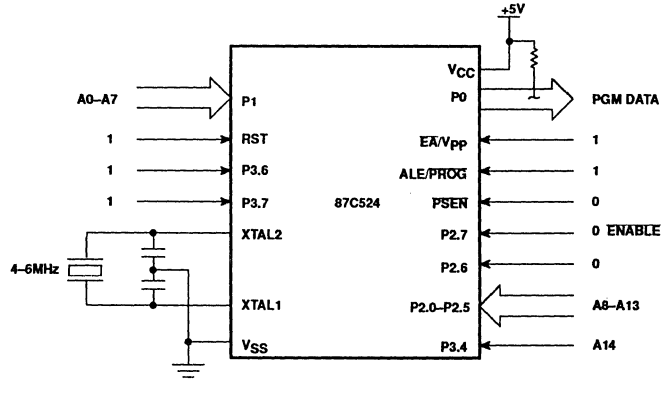
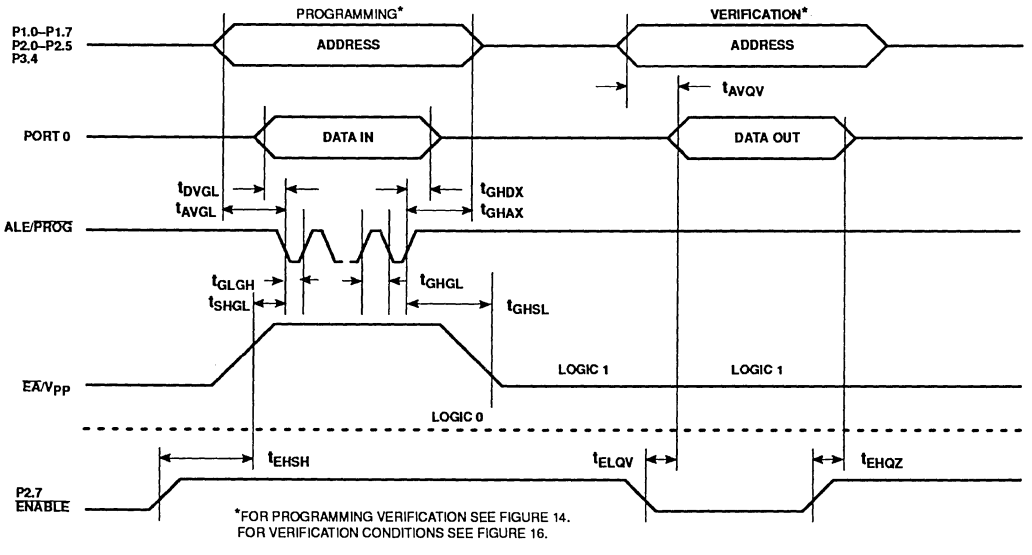


Figure 16. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_{amb} = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 17)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs



*FOR PROGRAMMING VERIFICATION SEE FIGURE 14.
FOR VERIFICATION CONDITIONS SEE FIGURE 16.

Figure 17. EPROM Programming and Verification

Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specifications defined by Philips.

CMOS single-chip 8-bit microcontroller

80C528/83C528/87C528

DESCRIPTION

The 8XC528 single-chip 8-bit microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 8XC528 has the same instruction set as the 80C51.

This device provides architectural enhancements that make it applicable in a variety of applications in consumer, telecom and general control systems, especially in those systems which need large ROM and RAM capacity on-chip.

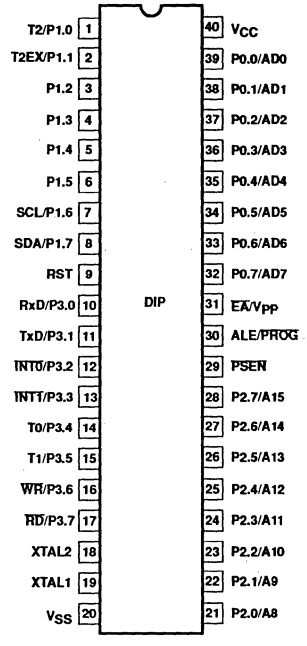
The 8XC528 contains a 32k × 8 ROM (83C528)/EPROM (87C528), a 512 × 8 RAM, four 8-bit I/O ports, two 16-bit timer/event counters (identical to the timers of the 80C51), a 16-bit timer (identical to the timer 2 of the 80C52), a watchdog timer with a separate oscillator, a multi-source, two-priority-level, nested interrupt structure, two serial interfaces (UART and I²C-bus), and on-chip oscillator and timing circuits.

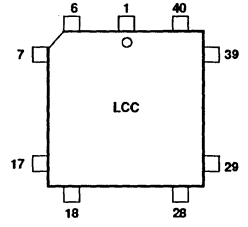
In addition, the 8XC528 has two software selectable modes of power reduction — idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

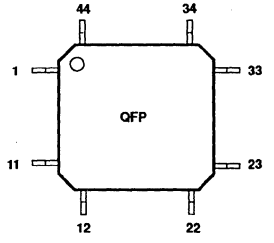
FEATURES

- 80C51 instruction set
  - 32k × 8 ROM (83C528)
  - 32k × 8 EPROM (87C528)
  - ROMless (80C528)
  - 512 × 8 RAM
  - Memory addressing capability 64k ROM and 64k RAM
  - Three 16-bit counter/timers
  - On-chip watchdog timer with oscillator
  - Full duplex UART
  - I²C serial interface
- Power control modes:
  - Idle mode
  - Power-down mode
  - Warm start from power-down
- CMOS and TTL compatible
- Two speed ranges at V_{CC} = 5V
  - 16MHz
  - 20MHz (87C528 only)
- Extended temperature ranges
- OTP package available
- ROM/EPROM code protection

PIN CONFIGURATIONS







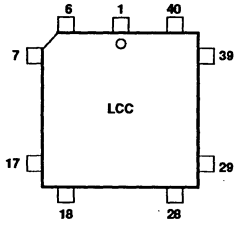
SEE PAGE 322 FOR LCC AND QFP PIN FUNCTIONS.

PART NUMBER SELECTION

PHILIPS PART ORDER NUMBER PART MARKING		SIGNETICS PART ORDER NUMBER			TEMPERATURE °C	
ROMless	ROM	ROMless	ROM	EPROM	AND PACKAGE	FREQ.
P80C528FBP	P83C528FBP/xxx	P80C528FBP N	P83C528FBP N	P87C528EBP N	0 to +70, Plastic DIP	16MHz
				P87C528EBF FA	0 to +70, Ceramic DIP with window	16MHz
P80C528FBA	P83C528FBA/xxx	P80C528FBA A	P83C528FBA A	P87C528EBA A	0 to +70, Plastic PLCC	16MHz
				P87C528EBL KA	0 to +70, Ceramic CLCC with window	16MHz
P80C528FBB	P83C528FBB/xxx	P80C528FBB B	P83C528FBB B	P87C528EBB B	0 to +70, Plastic QFP	16MHz
P80C528FFP	P83C528FFP/xxx	P80C528FFP N	P83C528FFP N	P87C528EFP N	–40 to +85, Plastic DIP	16MHz
				P87C528EFF FA	–40 to +85, Ceramic DIP with window	16MHz
P80C528FFA	P83C528FFA/xxx	P80C528FFA A	P83C528FFA A	P87C528EFA A	–40 to +85, Plastic PLCC	16MHz
				P87C528EFL KA	–40 to +85, Ceramic CLCC with window	16MHz
P80C528FFB	P83C528FFB/xxx	P80C528FFB B	P83C528FFB B	P87C528EFB B	–40 to +85, Plastic QFP	16MHz
P80C528FHP	P83C528FHP/xxx	P80C528FHP N	P83C528FHP N		–40 to +125, Plastic DIP	16MHz
					–40 to +125, Ceramic DIP with window	16MHz
P80C528FHA	P83C528FHA/xxx	P80C528FHA A	P83C528FHA A		–40 to +125, Plastic PLCC	16MHz
					–40 to +125, Ceramic CLCC with window	16MHz
P80C528FHB	P83C528FHB/xxx	P80C528FHB B	P83C528FHB B		–40 to +125, Plastic QFP	16MHz
				P87C528GBP N	0 to +70, Plastic DIP	20MHz
				P87C528GBF FA	0 to +70, Ceramic DIP with window	20MHz
				P87C528GBA A	0 to +70, Plastic PLCC	20MHz
				P87C528GBL KA	0 to +70, Ceramic CLCC with window	20MHz
				P87C528GBB B	0 to +70, Plastic QFP	20MHz
				P87C528GFP N	–40 to +85, Plastic DIP	20MHz
				P87C528GFF FA	–40 to +85, Ceramic DIP with window	20MHz
				P87C528GFA A	–40 to +85, Plastic PLCC	20MHz
				P87C528GFL KA	–40 to +85, Ceramic CLCC with window	20MHz
				P87C528GFB B	–40 to +85, Plastic QFP	20MHz

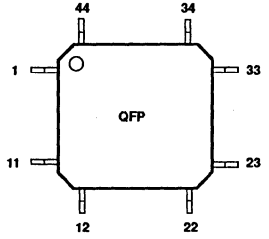
NOTE:
1. xxx denotes the ROM code number.

LCC PIN FUNCTIONS



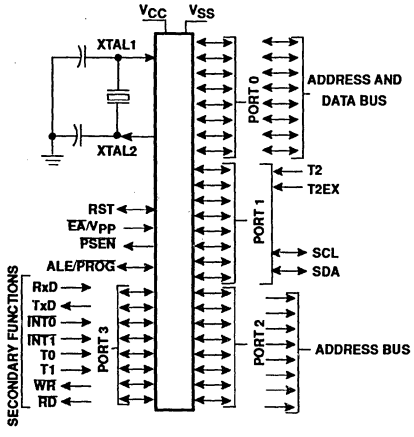
Pin	Function	Pin	Function
1	NC	23	NC
2	P1.0/T2	24	P2.0/A8
3	P1.1/T2EX	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6/SCL	30	P2.6/A14
9	P1.7/SDA	31	P2.7/A15
10	RST	32	$\overline{PSEN}$
11	P3.0/RxD	33	ALE/$\overline{PROG}$
12	NC	34	NC
13	P3.1/TxD	35	$\overline{EA}$/V_{PP}
14	P3.2/$\overline{INT0}$	36	P0.7/AD7
15	P3.3/$\overline{INT1}$	37	P0.6/AD6
16	P3.4/T0	38	P0.5/AD5
17	P3.5/T1	39	P0.4/AD4
18	P3.6/$\overline{WR}$	40	P0.3/AD3
19	P3.7/$\overline{RD}$	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	V_{SS}	44	V_{CC}

QFP PIN FUNCTIONS

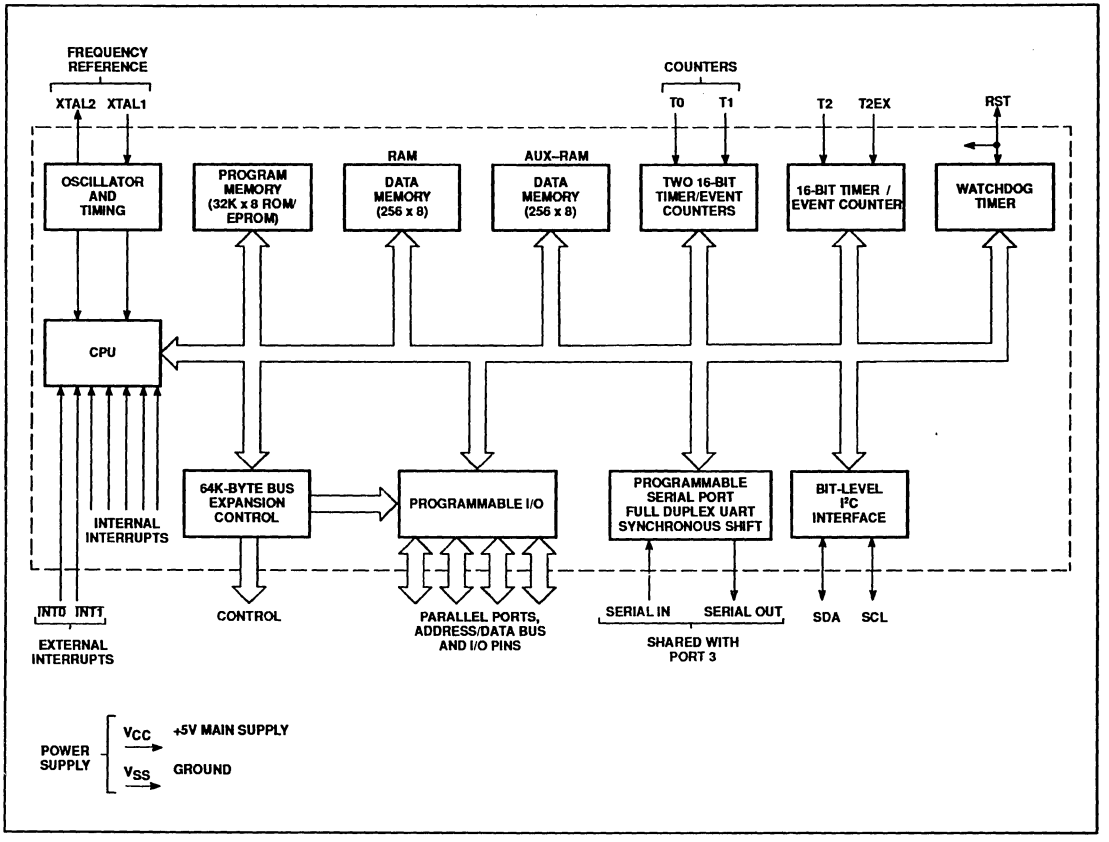


Pin	Function	Pin	Function
1	P1.5	23	P2.5/A13
2	P1.6/SCL	24	P2.6/A14
3	P1.7/SDA	25	P2.7/A15
4	RST	26	$\overline{PSEN}$
5	P3.0/RxD	27	ALE/$\overline{PROG}$
6	NC	28	NC
7	P3.1/TxD	29	$\overline{EA}$/V_{PP}
8	P3.2/$\overline{INT0}$	30	P0.7/AD7
9	P3.3/$\overline{INT1}$	31	P0.6/AD6
10	P3.4/T0	32	P0.5/AD5
11	P3.5/T1	33	P0.4/AD4
12	P3.6/$\overline{WR}$	34	P0.3/AD3
13	P3.7/$\overline{RD}$	35	P0.2/AD2
14	XTAL2	36	P0.1/AD1
15	XTAL1	37	P0.0/AD0
16	V_{SS}	38	V_{CC}
17	NC	39	NC
18	P2.0/A8	40	P1.0/T2
19	P2.1/A9	41	P1.1/T2EX
20	P2.2/A10	42	P1.2
21	P2.3/A11	43	P1.3
22	P2.4/A12	44	P1.4

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{SS}	20	22	16	I	**Ground:** 0V reference.
V_{CC}	40	44	38	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–36	37–30	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 also outputs the code bytes during program verification in the P87C528. External pull-ups are required during program verification.
P1.0–P1.7	1–8	2–9	40–44 1–3	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups, except P1.6 and P1.7 which have open drain. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 1 can sink/source one TTL (4 LSTTL) inputs. Port 1 receives the low-order address byte during program memory verification. Port 1 also serves alternate functions for timer 2:
	1	2	40	I	**T2 (P1.0):** Timer/counter 2 external count input.
	2	3	41	I	**T2EX (P1.1):** Timer/counter 2 trigger input.
	7	8	2	I/O	**SCL (P1.6):** I^2C serial port clock line.
	8	9	3	I/O	**SDA (P1.7):** I^2C serial port data line.
P2.0–P2.7	21–28	24–31	18–25	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	5, 7–13	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the SC80C51 family, as listed below:
	10	11	5	I	**RxD (P3.0):** Serial input port
	11	13	7	O	**TxD (P3.1):** Serial output port
	12	14	8	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	9	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	10	I	**T0 (P3.4):** Timer 0 external input
	15	17	11	I	**T1 (P3.5):** Timer 1 external input
	16	18	12	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	13	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	4	I/O	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}. After a watchdog timer overflow, this pin is pulled high while the internal reset signal is active.
ALE/$\overline{PROG}$	30	33	27	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{PSEN}$	29	32	26	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}/V_{PP}$	31	35	29	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 7FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 7FFFH. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	19	21	15	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	14	O	**Crystal 2:** Output from the inverting oscillator amplifier.

Table 1. Internal and External Program Memory Access with Security Bit Set

INSTRUCTION	ACCESS TO INTERNAL PROGRAM MEMORY	ACCESS TO EXTERNAL PROGRAM MEMORY
MOVC in internal program memory	YES	YES
MOVC in external program memory	NO	YES

ROM CODE PROTECTION

By setting a mask programmable security bit, the ROM content in the 83C528 is protected, i.e., it cannot be read out by any test mode or by any instruction in the external program memory space. The MOVC instructions are the only ones which have access to program code in the internal or external program memory. The $\overline{EA}$ input is latched during RESET and is 'don't care' after RESET. This implementation prevents reading from internal program code by switching from external program memory to internal program memory during MOVC instruction or an instruction that handles immediate data. Table 1 lists the access to the internal and external program memory by the MOVC instructions when the security bit has been set to logical one. If the security bit has been set to a logical 0 there are no restrictions for the MOVC instructions.

INTERNAL DATA MEMORY

The internal data memory is divided into three physically separated segments: 256 bytes of RAM, 256 bytes of AUX-RAM, and a 128 bytes special function area. These can be addressed each in a different way.

- RAM 0 to 127 can be addressed directly and indirectly as in the 80C51. Address pointers are R0 and R1 of the selected register bank.
- RAM 128 to 255 can only be addressed indirectly as in the 80C51. Address pointers are R0 and R1 of the selected register bank.
- AUX-RAM 0 to 255 is indirectly addressed in the same way as external data memory with the MOVX instructions. Address pointers are R0, R1 of the selected register bank and DPTR. An access to AUX-RAM 0 to 255 will not affect ports P0, P2, P3.6 and P3.7.

An access to external data memory locations higher than 255 will be performed with the MOVX DPTR instructions in the same way as in the 8051 structure, so with P0 and P2 as data/address bus and P3.6 and P3.7 as write and read timing signals. Note that these external data memory cannot be accessed with R0 and R1 as address pointer.

TIMER 2

Timer 2 is functionally equal to the Timer 2 of the 8052AH. Timer 2 is a 16-bit timer/counter. These 16 bits are formed by two special function registers TL2 and TH2. Another pair of special function register RCAP2L and RCAP2H form a 16-bit capture register or a 16-bit reload register. Like Timer 0 and 1, it can operate either as a timer or as an event counter. This is selected by bit C/T2N in the special function register T2CON. It has three operating modes: capture, autoload, and baud rate generator mode which are selected by bits in T2CON.

WATCHDOG TIMER T3

The watchdog timer consists of an 11-bit prescaler and an 8-bit timer formed by special function register T3. The prescaler is incremented by an on-chip oscillator with a fixed frequency of 1MHz. The maximum tolerance on this frequency is –50% and +100%. The 8-bit timer increments every 2048 cycles of the on-chip oscillator. When a timer overflow occurs, the microcontroller is reset and a reset output pulse of 16 × 2048 cycles of the on-chip oscillator is generated at pin RST. The internal RESET signal is not inhibited when the external RST pin is kept low by, for example, an external reset circuit. The RESET signal drives port 1, 2, 3 into the high state and port 0 into the high impedance state.

The watchdog timer is controlled by one special function register WDCON with the direct address location A5H. WDCON can be read and written by software. A value of A5H in WDCON halts the on-chip oscillator and clears both the prescaler and timer T3. After the RESET signal, WDCON contains A5H. Every value other than A5H in WDCON enables the watchdog timer. When the watchdog timer is enabled, it runs independently of the XTAL-clock.

Timer T3 can be read on the fly. Timer T3 can only be written if WDCON contains the value 5AH. A successful write operation to T3 will clear the prescaler and WDCON, leaving the watchdog enabled and preventing inadvertent changes of T3. To prevent an overflow of the watchdog timer, the user program has to reload the watchdog timer within periods that are shorter than the programmed watchdog timer internal. This time interval is determined by an 8-bit value that has to be loaded in register T3 while at the same time the prescaler is cleared by hardware.

Watchdog timer interval =

$$\frac{[256-(T3)] \times 2048}{\text{on-chip oscillator frequency}}$$

BIT-LEVEL I²C INTERFACE

This bit-level serial I/O interface supports the I²C-bus. P1.6/SCL and P1.7/SDA are the serial I/O pins. These two pins meet the I²C specification concerning the input levels and output drive capability. Consequently, these pins have an open drain output configuration. All the four modes of the I²C-bus are supported:

- master transmitter
- master receiver
- slave transmitter
- slave receiver

The advantages of the bit-level I²C hardware compared with a full software I²C implementation are:

- the hardware can generate the SCL pulse
- Testing a single bit (RBF respectively, WBF) is sufficient as a check for error free transmission.

The bit-level I²C hardware operates on serial bit level and performs the following functions:

- filtering the incoming serial data and clock signals
- recognizing the START condition
- generating a serial interrupt request SI after reception of a START condition and the first falling edge of the serial clock
- recognizing the STOP condition
- recognizing a serial clock pulse on the SCL line
- latching a serial bit on the SDA line (SDI)
- stretching the SCL LOW period of the serial clock to suspend the transfer of the next serial data bit
- setting Read Bit Finished (RBF) when the SCL clock pulse has finished and Write Bit

Finished (WBF) if there is no arbitration loss detected (i.e., SDA = 0 while SDO = 1)
- setting a serial clock Low-to-High detected (CLH) flag
- setting a Bus Busy (BB) flag on a START condition and clearing this flag on a STOP condition
- releasing the SCL line and clearing the CLH, RBF and WBF flags to resume transfer of the next serial data bit
- generating an automatic clock if the single bit data register S1BIT is used in master mode.

The following functions must be done in software:
- handling the I²C START interrupts
- converting serial to parallel data when receiving
- converting parallel to serial data when transmitting
- comparing the received slave address with its own
- interpreting the acknowledge information
- guarding the I²C status if RBF or WBF = 0.

Additionally, if acting as master:
- generating START and STOP conditions
- handling bus arbitration
- generating serial clock pulses if S1BIT is not used.

Three SFRs control the bit-level I²C interface: S1INT, S1BIT and S1SCS.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the Logic Symbol.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-up reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-up, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. The power-down mode can be terminated by a RESET in the same way as in the 80C51 or in addition by one of two external interrupts, INT0 or INT1. A termination with an external interrupt does not affect the internal data memory and does not affect the special function registers. This makes it possible to exit power-down without changing the port output levels. To terminate the power-down mode with an external interrupt $\overline{INT0}$ or $\overline{INT1}$ must be switched to level-sensitive and must be enabled. The external interrupt input signal $\overline{INT0}$ and $\overline{INT1}$ must be kept low until the oscillator has restarted and stabilized. An instruction following the instruction that puts the device in the power-down mode will be executed. A reset generated by the watchdog timer terminates the power-down mode in the same way as an external RESET, and only the contents of the on-chip RAM are preserved. The control bits for the reduced power modes are in the special function register PCON.

DESIGN CONSIDERATIONS

At power-on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

When the idle mode is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write when idle is terminated by reset, the instruction following the one that invokes idle should not be one that writes to a port pin or to external memory.

Table 2 shows the state of I/O ports during low current operating modes.

Table 2. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Operating temperature under bias	0 to +70, or –40 to +85, or –40 to +125	°C
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ pin to V_{SS}	0 to +13.0	V
Voltage on any other pin to V_{SS}	–0.5 to V_{CC} +0.5	V
Input, output current on any two pins	±10	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.0	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS (80C528/83C528)

T_{amb} = 0°C to +70°C (V_{CC} = 5V ±20%), –40°C to +85°C (V_{CC} = 5V ±20%), or –40°C to +125°C (V_{CC} = 5V ±10%), V_{SS}=0V

SYMBOL	PARAMETER	PART TYPE	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{IL}	Input low voltage, except $\overline{EA}$, P1.6/SCL, P1.7/SDA	0°C to 70°C –40°C to +85°C –40°C to +125°C		–0.5 –0.5 –0.5	0.2V_{CC}–0.1 0.2V_{CC}–0.15 0.2V_{CC}–0.25	V V V
V_{IL1}	Input low voltage to $\overline{EA}$	0°C to 70°C –40°C to +85°C –40°C to +125°C		–0.5 –0.5 –0.5	0.2V_{CC}–0.3 0.2V_{CC}–0.35 0.2V_{CC}–0.45	V V V
V_{IL2}	Input low voltage to P1.6/SCL, P1.7/SDA[3]			–0.5	0.3V_{CC}	V
V_{IH}	Input high voltage, except XTAL1, RST, P1.6/SCL, P1.7/SDA	0°C to 70°C –40°C to +85°C –40°C to +125°C		0.2V_{CC}+0.9 0.2V_{CC}+1.0 0.2V_{CC}+1.0	V_{CC}+0.5 V_{CC}+0.5 V_{CC}+0.5	V V V
V_{IH1}	Input high voltage, XTAL1, RST	0°C to 70°C –40°C to +85°C –40°C to +125°C		0.7V_{CC} 0.7V_{CC}+0.1 0.7V_{CC}+0.1	V_{CC}+0.5 V_{CC}+0.5 V_{CC}+0.5	V V V
V_{IH2}	Input high voltage, P1.6/SCL, P1.7/SDA[3]			0.7V_{CC}	6.0	V
V_{OL}	Output low voltage, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA[1]		I_{OL} = 1.6mA[4]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$[1]		I_{OL} = 3.2mA[4]		0.45	V
V_{OL2}	Output low voltage, P1.6/SCL, P1.7/SDA		I_{OL} = 3.0mA[4]		0.4	V
V_{OH}	Output high voltage, ports 1, 2, 3		I_{OH} = –60µA, I_{OH} = –25µA I_{OH} = –10µA	2.4 0.75V_{CC} 0.9V_{CC}		V V V
V_{OH1}	Output high voltage, Port 0 in external bus mode, ALE, $\overline{PSEN}$, RST[2]		I_{OH} = –800µA, I_{OH} = –300µA I_{OH} = –80µA	2.4 0.75V_{CC} 0.9V_{CC}		V V V
I_{IL}	Logical 0 input current, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA	0°C to 70°C –40°C to +85°C –40°C to +125°C	V_{IN} = 0.45V		–50 –75 –75	µA µA µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA	0°C to 70°C –40°C to +85°C –40°C to +125°C	See note 5		–650 –750 –750	µA µA µA

DC ELECTRICAL CHARACTERISTICS (80C528/83C528) (Continued)

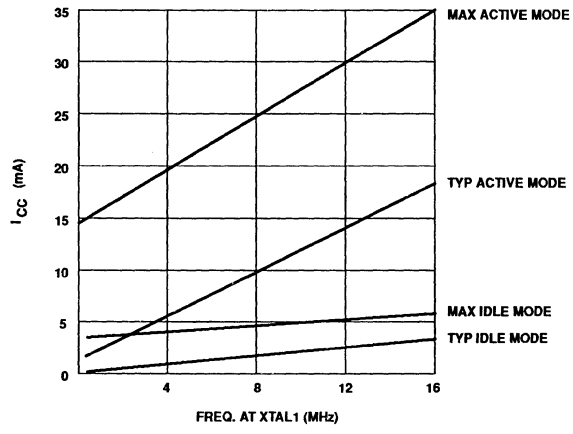
T_{amb} = 0°C to +70°C (V_{CC} = 5V ±20%), –40°C to +85°C (V_{CC} = 5V ±20%), or –40°C to +125°C (V_{CC} = 5V ±10%), V_{SS}=0V

SYMBOL	PARAMETER	PART TYPE	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
I_{IL1}	Input leakage current, port 0		0.45<Vi<V_{CC}		±10	µA
I_{IL2}	Input leakage current, P1.6/SCL, P1.7/SDA		0V<Vi<6.0V 0V<V_{CC}<6.0V		±10	µA µA
I_{CC}	Power supply current: Active mode @ 16MHz Idle mode @ 16MHz Power down mode Power down mode	 –40°C to +125°C	See notes 6, 7		 35 6 100 150	 mA mA µA µA
R_{RST}	Internal reset pull-down resistor			50	150	kΩ
C_{IO}	Pin capacitance		Freq.=1MHz		10	pF

NOTES:

1. Capacitive loading on Port 0 and Port 2 may cause spurious noise pulses to be superimposed on the LOW level ouput voltage of ALE, Port 1 and Port 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make a 1-to-0 transition during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE line may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
2. Capacitive loading on Port 0 and Port 2 may cause the HIGH level output voltage on ALE and $\overline{PSEN}$ to momentarily fall below the 0.9V_{CC} specification when the address bits are stabilizing.
3. The input threshold voltage of P1.6 and P1.7 (SIO1) meets the I^2C specification, so a voltage below 0.3V_{CC} will be recognized as a logic 0 while an input above 0.7V_{CC} will be recognized as a logic 1.
4. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 10mA
   Maximum I_{OL} per 8–bit port: –
   Port 0: 26mA
   Ports 1, 2, & 3: 15mA
   Maximum total I_{OL} for all output pins: 71mA

   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
5. Pins of ports 1, 2, and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
6. See Figures 9 through 12 for I_{CC} test conditions.
7. I_{CCMAX} at other frequencies can be derived from the figure below, where FREQ is the external oscillator frequency in MHz. I_{CCMAX} is given in mA.



VALID ONLY WITHIN FREQUENCY SPECIFICATIONS OF DEVICE UNDER TEST.

I_{CC} vs. FREQUENCY

DC ELECTRICAL CHARACTERISTICS (87C528)

T_{amb} = 0°C to +70°C, or –40°C to +85°C, V_{CC} = 5V±10%, V_{SS}=0V

SYMBOL	PARAMETER	PART TYPE	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{IL}	Input low voltage, except $\overline{EA}$, P1.6/SCL, P1.7/SDA	0°C to 70°C –40°C to +85°C		–0.5 –0.5	0.2V_{CC}–0.1 0.2V_{CC}–0.15	V V
V_{IL1}	Input low voltage to $\overline{EA}$	0°C to 70°C –40°C to +85°C		0 0	0.2V_{CC}–0.3 0.2V_{CC}–0.35	V V
V_{IL2}	Input low voltage to P1.6/SCL, P1.7/SDA[5]			–0.5	1.5	V
V_{IH}	Input high voltage, except XTAL1, RST, P1.6/SCL, P1.7/SDA	0°C to 70°C –40°C to +85°C		0.2V_{CC}+0.9 0.2V_{CC}+1.0	V_{CC}+0.5 V_{CC}+0.5	V V
V_{IH1}	Input high voltage, XTAL1, RST	0°C to 70°C –40°C to +85°C		0.7V_{CC} 0.7V_{CC}+0.1	V_{CC}+0.5 V_{CC}+0.5	V V
V_{IH2}	Input high voltage, P1.6/SCL, P1.7/SDA[5]			3.0	6.0	V
V_{OL}	Output low voltage, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA		I_{OL} = 1.6mA [1]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$		I_{OL} = 3.2mA [1]		0.45	V
V_{OL2}	Output low voltage, P1.6/SCL, P1.7/SDA		I_{OL} = 3.0mA [1]		0.4	V
V_{OH}	Output high voltage, ports 1, 2, 3 [2]		I_{OH} = –60µA, I_{OH} = –25µA	2.4 0.75V_{CC}		V V
V_{OH1}	Output high voltage, Port 0 in external bus mode, ALE, $\overline{PSEN}$, RST		I_{OH} = –800µA, I_{OH} = –300µA	2.4 0.75V_{CC}		V V
I_{IL}	Logical 0 input current, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA	0°C to 70°C –40°C to +85°C	V_{IN} = 0.45V		–50 –75	µA V
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA	0°C to 70°C –40°C to +85°C	See note 3		–650 –750	µA µA
I_{L1}	Input leakage current, port 0		V_{IN} = V_{IL} or V_{IH}		±10	µA
I_{L2}	Input leakage current, P1.6/SCL, P1.7/SDA		0V <Vi < 6.0V 0V <V_{CC} < 6.0V		±10	µA
I_{CC}	Power supply current: Active mode @ 16MHz Idle mode @ 16MHz Power down mode	 0°C to 70°C –40°C to +85°C 0°C to 70°C –40°C to +85°C	See note 4		 25 35 5 6 50	 mA mA mA mA µA
R_{RST}	Internal reset pull-down resistor			50	300	kΩ
C_{IO}	Pin capacitance				10	pF

NOTES:

1. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows: 10mA per port pin, port 0 total (all bits) 26mA, ports 1, 2, and total each (all bits) 15mA.
2. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the 0.9V_{CC} specification when the address bits are stabilizing.
3. Pins of ports 1, 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
4. See Figures 10 through 13 for I_{CC} test conditions.
5. The input threshold voltage of P1.6 and P1.7 (SIO1) meets the I^2C specification, so an input voltage below 1.5V will be recognized as a logic 0 while an input voltage above 3.0V will be recognized as a logic 1.

AC ELECTRICAL CHARACTERISTICS[1, 2]

SYMBOL	FIGURE	PARAMETER	16MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	1	Oscillator frequency: **Speed Versions** 80C528 ALL 83C528 ALL 87C528 P87C528EXX P87C528GXX			 1.2 1.2 3.5 3.5	 16 16 16 20	 MHz MHz MHz MHz
t_{LHLL}	1	ALE pulse width	85		$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	8		$t_{CLCL}-55$		ns
t_{LLAX}	1	Address hold after ALE low	28		$t_{CLCL}-35$		ns
t_{LLIV}	1	ALE low to valid instruction in		150		$4t_{CLCL}-100$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	23		$t_{CLCL}-40$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	143		$3t_{CLCL}-45$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		83		$3t_{CLCL}-105$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		38		$t_{CLCL}-25$	ns
t_{AVIV}	1	Address to valid instruction in		208		$5t_{CLCL}-105$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	275		$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	275		$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		148		$5t_{CLCL}-165$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		55		$2t_{CLCL}-70$	ns
t_{LLDV}	2, 3	ALE low to valid data in		350		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		398		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	138	238	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	120		$4t_{CLCL}-130$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	3		$t_{CLCL}-60$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	13		$t_{CLCL}-50$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	23	103	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	6	High time	20		20		ns
t_{CLCX}	6	Low time	20		20		ns
t_{CLCH}	6	Rise time		20		20	ns
t_{CHCL}	6	Fall time		20		20	ns
Shift Register							
t_{XLXL}	4	Serial port clock cycle time	750		$12t_{CLCL}$		ns
t_{QVXH}	4	Output data setup to clock rising edge	492		$10t_{CLCL}-133$		ns
t_{XHQX}	4	Output data hold after clock rising edge	8		$2t_{CLCL}-117$		ns
t_{XHDX}	4	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	4	Clock rising edge to input data valid		492		$10t_{CLCL}-133$	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.

AC ELECTRICAL CHARACTERISTICS – I²C INTERFACE

SYMBOL	PARAMETER	INPUT	OUTPUT	I²C SPECIFICATION
SCL TIMING CHARACTERISTICS				
t_{HD}; STA	START condition hold time	≥ 14 t_{CLCL}[1]	Note 2	≥ 4.0µs
t_{LOW}	SCL LOW time	≥ 16 t_{CLCL}	Note 2	≥ 4.7µs
t_{HIGH}	SCL HIGH time	≥ 14 t_{CLCL}[1]	≥ 80 t_{CLCL}[3]	≥ 4.0µs
t_{RC}	SCL rise time	≤ 1µs[4]	Note 5	≤ 1.0µs
t_{FC}	SCL fall time	≤ 0.3µs[4]	≤ 0.3µs [6]	≤ 0.3µs
SDA TIMING CHARACTERISTICS				
t_{SU}; DAT1	Data set-up time	≥ 250ns	Note 2	≥ 250ns
t_{HD}; DAT	Data hold time	≥ 0ns	Note 2	≥ 0ns
t_{SU}; STA	Repeated START set-up time	≥ 14 t_{CLCL}[1]	Note 2	≥ 4.7µs
t_{SU}; STO	STOP condition set-up time	≥ 14 t_{CLCL}[1]	Note 2	≥ 4.0µs
t_{BUF}	Bus free time	≥ 14 t_{CLCL}[1]	Note 2	≥ 4.7µs
t_{RD}	SDA rise time	≤ 1µs[4]	Note 5	≤ 1.0µs
t_{FD}	SDA fall time	≤ 0.3µs[4]	≤ 0.3µs [6]	≤ 0.3µs

NOTES:
1. At f_{CLK} = 3.5MHz, this evaluates to 14 × 286ns = 4µs, i.e., the bit-level I²C interface can respond to the I²C protocol for f_{CLK} ≥ 3.5MHz.
2. This parameter is determined by the user software, it has to comply with the I²C.
3. This value gives the autoclock pulse length which meets the I²C specification for the specified XTAL clock frequency range. Alternatively, the SCL pulse may be timed by software.
4. Spikes on SDA and SCL lines with a duration of less than 4 × f_{CLK} will be filtered out.
5. The rise time is determined by the external bus line capacitance and pull-up resistor, it must be ≤ 1µs.
6. The maximum capacitance on bus lines SDA and SCL is 400pF.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

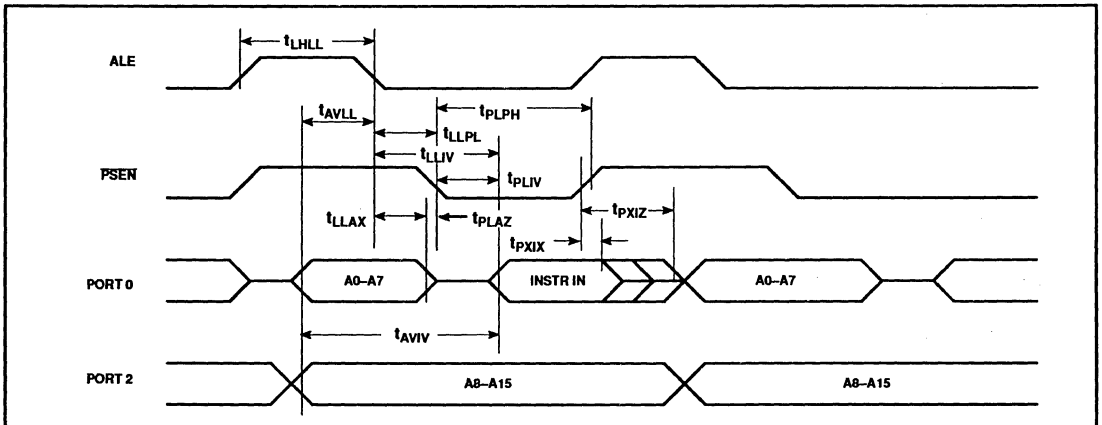


Figure 1. External Program Memory Read Cycle

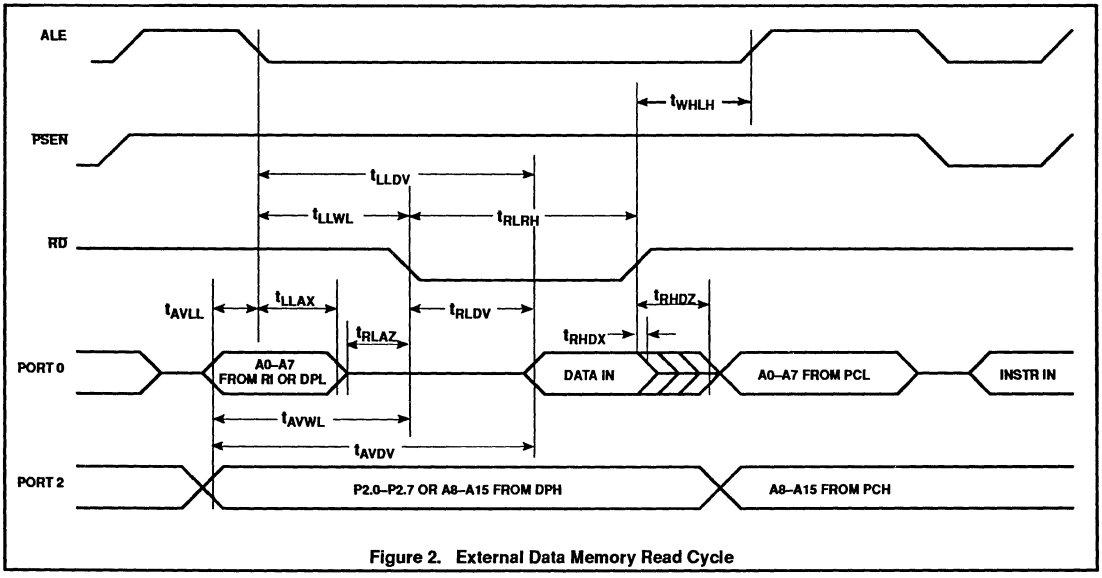


Figure 2. External Data Memory Read Cycle

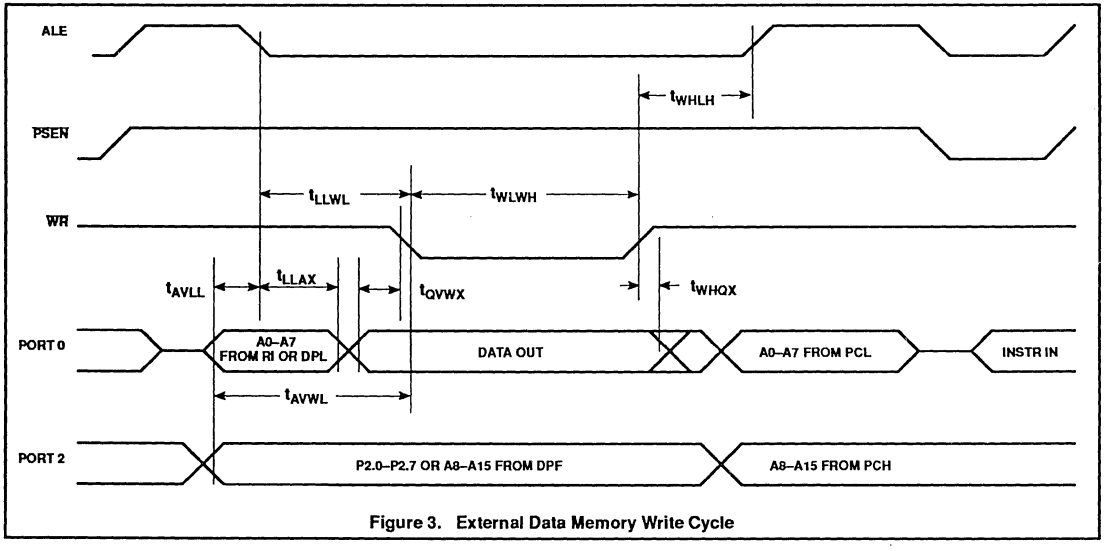


Figure 3. External Data Memory Write Cycle

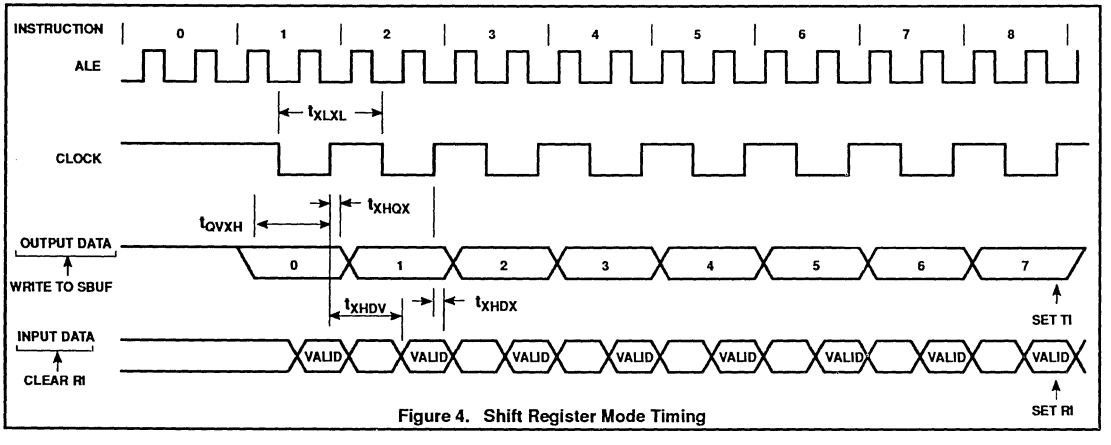


Figure 4. Shift Register Mode Timing

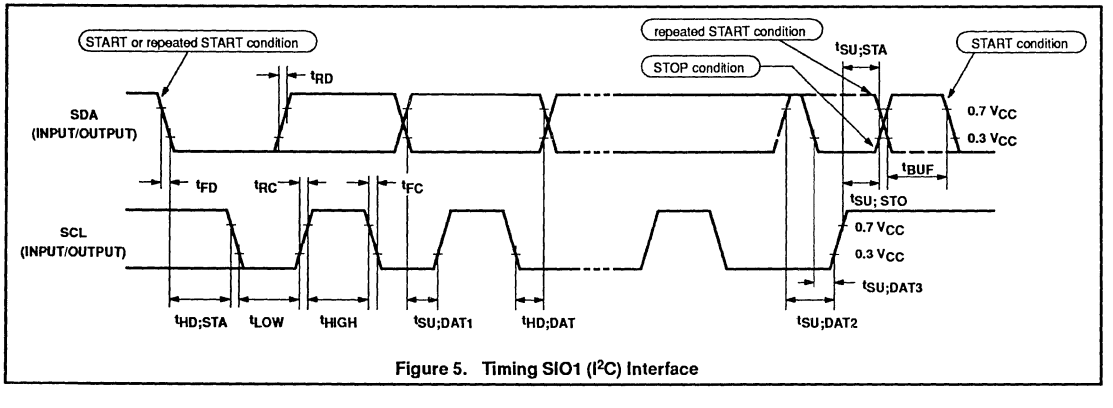


Figure 5. Timing SIO1 (I^2C) Interface

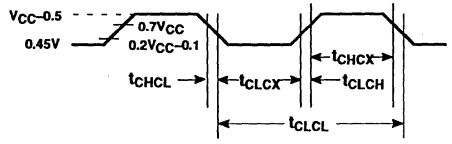
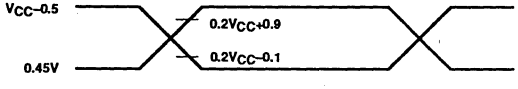
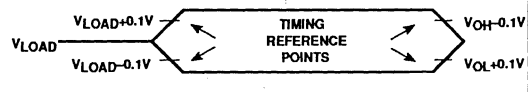


Figure 6. External Clock Drive



NOTE:
AC inputs during testing are driven at V_{CC} –0.5 for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at V_{IH} min for a logic '1' and V_{IL} for a logic '0'.

Figure 7. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 8. Float Waveform

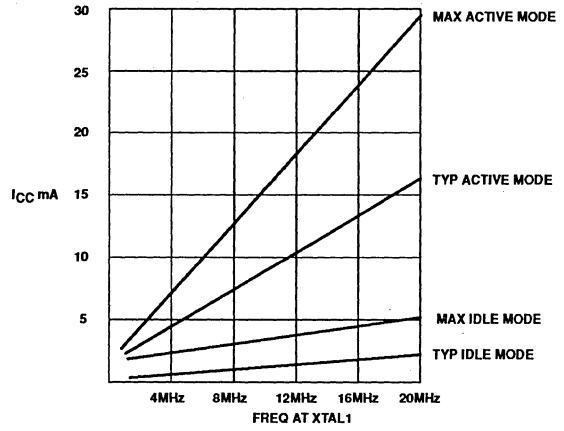


Figure 9. I_{CC} vs. FREQ (87C528)
Valid only within frequency specifications of the device under test

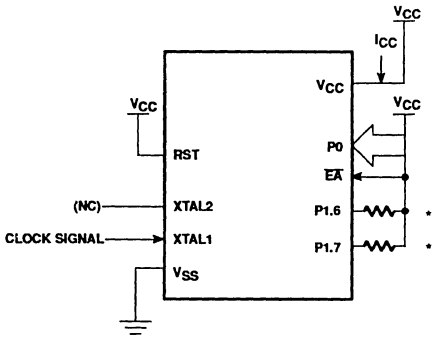


Figure 10. I_{CC} Test Condition, Active Mode
All other pins are disconnected

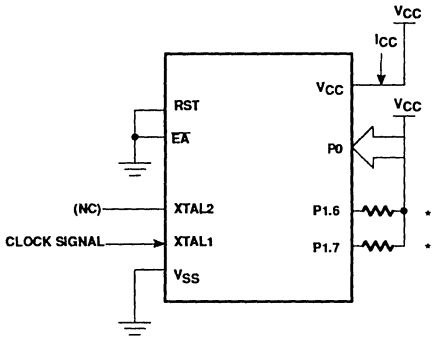


Figure 11. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

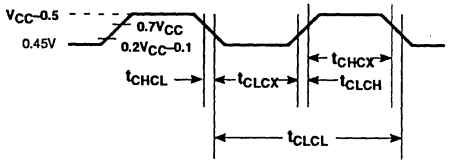


Figure 12. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

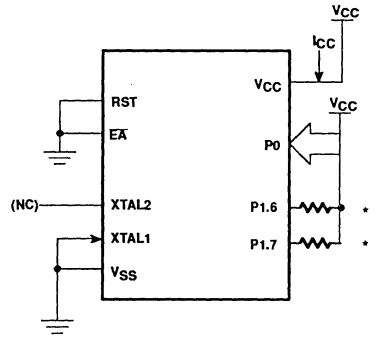


Figure 13. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

NOTE:
* Ports 1.6 and 1.6 should be connected to V_{CC} through resistors of sufficiently high value such that the sink current into these pins does not exceed the I_{OL1} specifications.

EPROM CHARACTERISTICS

The 87C528 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C528 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C528 manufactured by Philips.

Table 3 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 14 and 15. Figure 16 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 14. Note that the 87C528 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1, 2 and 3, as shown in Figure 14. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 2 and 3 specified in Table 3 are held at the 'Program Code Data' levels indicated in Table 3. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 15.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 3FH, using the'"Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25 pulse programming sequence using the 'Pgm Lock Bit' levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 1, 2 and 3 as shown in Figure 16. The other pins are held at the 'Verify Code Data' levels indicated in Table 3. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(031H) = 97H indicates 87C528

Program Lock Bits

The 87C524 has 3 programmable lock bits that will provide different levels of protection for the on-chip code and data (see Table 4).

Erasing the EPROM also erases the encryption array and the program lock bits, returning the part to full functionality.

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 3, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

™Trademark phrase of Intel Corporation.

Table 3. EPROM Progamming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0
Pgm lock bit 3	1	0	0*	V_{PP}	0	1	0	1

NOTES:
1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.
* ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

Table 4.

PROGRAM LOCK BITS[1,2]				
	LB1	LB2	LB3	PROTECTION DESCRIPTION
1	U	U	U	No Program Lock features enabled. (Code verify will still be encrypted by the Encryption Array if programmed.)
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is jumped and latched on Reset, and further programming of the EPROM is disabled.
3	P	P	U	Same as 2, also verify is disabled.
4	P	P	P	Same as 3, external execution is disabled. Internal data RAM is not accessable.

NOTES:
1. P – programmed. U – unprogrammed.
2. Any other combination of the lock bits is not defined.

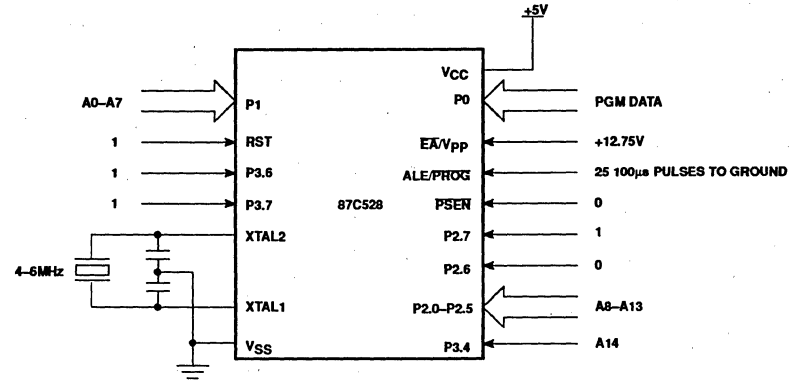


Figure 14. Programming Configuration

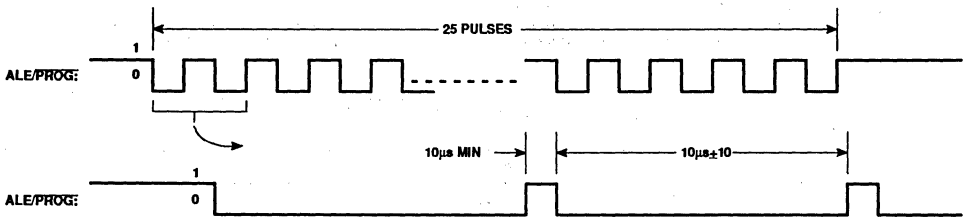


Figure 15. PROG Waveform

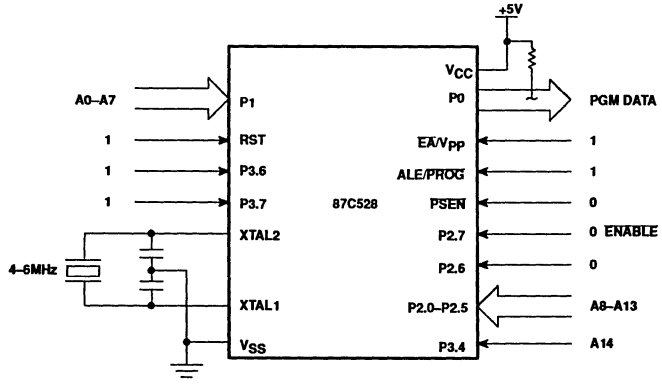


Figure 16. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_{amb} = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 17)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

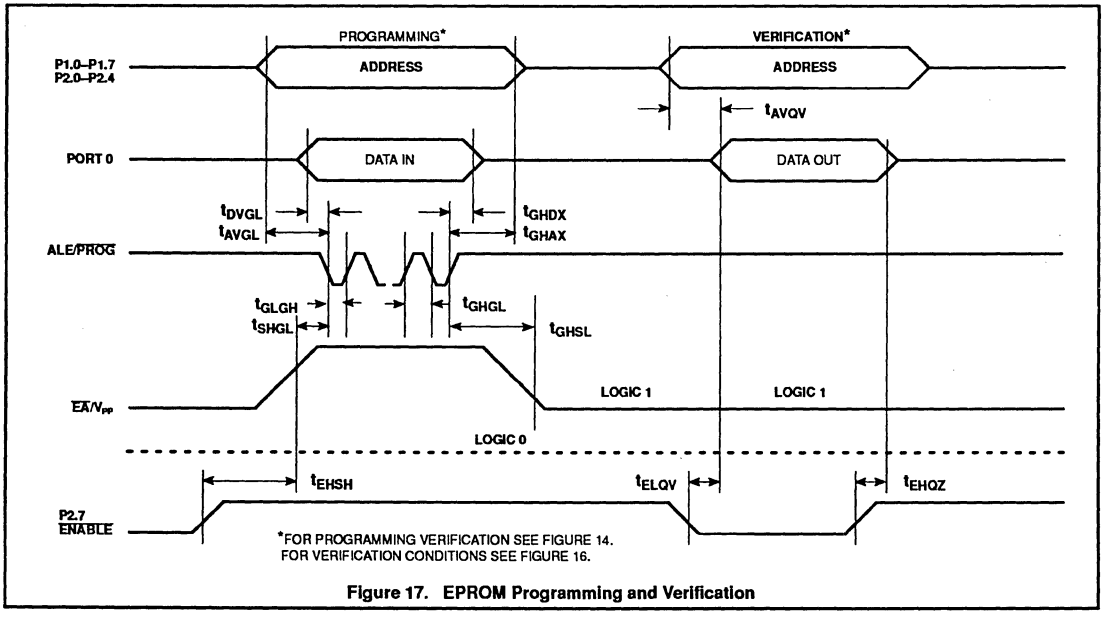


Figure 17. EPROM Programming and Verification

Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specifications defined by Philips.

8XC550 OVERVIEW

The Signetics 8XC550 is a single-chip control oriented microcontroller in the 80C51 family. The 8XC550 has the same basic architecture and instruction set as the industry standard 80C51, with the addition of an eight-channel multiplexed 8-bit A/D converter and a versatile watchdog timer.

The part is available in three versions, collectively referred to as the 8XC550: the 83C550 with 4k bytes of masked ROM program memory; the 87C550 with 4k bytes of EPROM program memory; and the 80C550 with no on-chip program memory. The EPROM version of this device is available in both quartz-lid erasable and one-time programmable (OTP) packages. Once the EPROM array of the 87C550 is programmed, the part will generally be functionally equivalent to the masked ROM 83C550. The watchdog timer setup, however, is accomplished in a different manner on the ROM part than it is on the EPROM and ROMless parts. Refer to the description of the watchdog timer for details.

The 8XC550 supports two power reduction modes of operation referred to as the idle mode and the power-down mode. The 8XC550 features include:

- 80C51 based architecture
- Eight-channel multiplexed 8-bit A/D converter in the LCC package (six channels on the DIP package)
- 40-microsecond A/D conversion time (when used with a 12MHz oscillator)
- Separate A/D power supply and references (LCC part only)
- Watchdog timer
- 4k byte ROM or EPROM program memory
- 128-byte RAM
- 40-pin DIP and 44-pin LCC packages
- 24 I/O lines + 8 input only lines on the LCC package (24 I/O lines + 6 input only lines on the DIP package), port 1 is input only
- Two 16-bit counter/timers
- Two external interrupts
- External memory addressing capability of 64k program memory and 64k data memory
- Full duplex UART
- Low power consumption
  - Idle mode
  - Power-down mode

Differences From the 80C51

Special Function Registers

The 8XC550 contains all of the special function registers (SFRs) that are present on the standard 80C51. There are several SFRs that have been added as well as several others that have been modified somewhat in order to accommodate the additional functions of this chip.

Table 1 contains a summary of all of the SFRs and their addresses.

The A/D control register (ADCON) and the A/D data register (ADAT) have been added to allow access to the on-chip A/D converter. ADCON contains all of the control and status bits required to operate the A/D converter.

Four other registers have been added to allow control of the watchdog timer function. The WDCON register controls watchdog timer operation, and WDL holds the watchdog timer reload value. The WFEED1 and WFEED2 registers, when properly manipulated, cause the watchdog timer to be reloaded from WDL. The current watchdog counter value may also be read back from the WDL register address.

I/O Port Structure

The 8XC550 has the same I/O ports as the standard 80C51 with the following exceptions: the port 1 pins are shared with the A/D converter inputs and are input only pins even when the A/D is not being used; port 1 has only 6 pins on the DIP version of the part.

A/D Converter

The LCC packaged versions of the 8XC550 contain an eight-channel multiplexed 8-bit A/D converter, while the DIP versions implement only six channels. The conversion requires 40 machine cycles (40µs at 12MHz oscillator frequency).

The A/D converter is controlled by the A/D control register, ADCON. Input channels are selected by the analog multiplexer by bits ADCON.0 through ADCON.2. The ADCON register is not bit addressable.

The completion of the 8-bit ADC conversion is flagged by ADCI in the ADCON register, and the result is stored in the special function register ADAT.

An ADC conversion in progress is unaffected by an ADC start. The result of a completed conversion remains unaffected provided ADCI remains at a logic 1. While ADCS is a logic 1 or ADCI is a logic 1, a new ADC START will be blocked and consequently lost. An ADC conversion in progress is aborted when the idle or power-down mode is entered. The result of a completed conversion (ADCI = logic 1) remains unaffected when entering the idle mode. See Figure 1 for an A/D input equivalent circuit.

The analog input pins ADC0-ADC7 may still be used as digital inputs. The analog input channel that is selected by the ADDR2-ADDR0 bits in ADCON cannot be used as a digital input. Reading the selected A/D channel as a digital input will always return a 1. The unselected A/D inputs may always be used as digital inputs.

Table 1. 8XC550 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
ADAT#	A/D result	C6H									xxH
ADCON#	A/D control	C5H	–	–	–	ADCI	ADCS	AADR2	AADR1	AADR0	xxx00000B
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes):										
DPH	High byte	83H									00H
DPL	Low byte	82H									00H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*#	Interrupt priority	B8H	–	PWD	PAD	PS	PT1	PX1	PT0	PX0	x0000000B
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*#	Interrupt enable	A8H	EA	EWD	EAD	ES	ET1	EX1	ET0	EX0	00H
P0*	Port 0	80H	87	86	85	84	83	82	81	80	FFH
P1*	Port 1	90H	97	96	95	94	93	92	91	90	FFH
P2*	Port 2	A0H	A7	A6	A5	A4	A3	A2	A1	A0	FFH
P3*	Port 3	B0H	B7	B6	B5	B4	B3	B2	B1	B 0	FFH
PCON#	Power control	87H	SMOD	SIDL	–	–	GF1	GF0	PD	IDL	00xx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
SBUF	Serial data buffer	99H									xxH
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial port control	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	00H
TCON*	Timer counter/control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
TMOD	Timer/counter mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
TH0	Timer 0 high byte	8CH									00H
TH1	Timer 1 high byte	8DH									00H
TL0	Timer 0 low byte	8AH									00H
TL1	Timer 1 low byte	8BH									00H
			C7	C6	C5	C4	C3	C2	C1	C0	
WDCON*#	Watchdog timer control	C0H	PRE2	PRE1	PRE0	–	–	WDRUN	WDTOF	WDMOD	000xx000B**
WDL#	Watchdog timer reload	C1H									FFH**
WFEED1#	Watchdog timer feed 1	C2H									xxH
WFEED2#	Watchdog timer feed 2	C3H									xxH

* SFRs are bit addressable.
# SFRs are modified from or added to the 80C51 SFRs.
**This value is not valid for a masked ROM part (83C550) when running from internal memory ($\overline{EA}$ = 1). See data sheet for details.

ADCON Register

MSB							LSB
X	X	X	ADCI	ADCS	AADR2	AADR1	AADR0

ADCI	ADCS	Operation
0	0	ADC not busy, a conversion can be started.
0	1	ADC busy, start of a new conversion is blocked.
1	0	Conversion completed, start of a new is blocked.
1	1	Not possible.

INPUT CHANNEL SELECTION			
ADDR2	ADDR1	ADDR0	INPUT PIN
0	0	0	P1.0
0	0	1	P1.1
0	1	0	P1.2
0	1	1	P1.3
1	0	0	P1.4
1	0	1	P1.5
1	1	0	P1.6*
1	1	1	P1.7*

*Not present on 40-pin DIP versions.

Symbol	Position	Function
ADCI	ADCON.4	ADC interrupt flag. This flag is set when an ADC conversion is complete. If IE.5 = 1, an interrupt is requested when ADCI = 1. The ADCI flag must be cleared by software after A/D data is read, before the next conversion can begin.
ADCS	ADCON.3	ADC start and status. Setting this bit starts an A/D conversion. Once set, ADCS remains high throughout the conversion cycle. On completion of the conversion, it is reset at the same time the ADCI interrupt flag is set. ADCS cannot be reset by software.
AADR2	ADCON.2	Analog input selects.
AADR1	ADCON.1	Binary coded address
AADR0	ADCON.0	selects one of the five analog input port pins of P1 to be input to the converter. It can only be changed when ADCI and ADCS are both low. AADR2 is the most significant bit.

Sample A/D Routines

The following routines demonstrate two methods of operating the A/D converter. The first method uses polling to determine when the A/D conversion is complete. The second method uses the A/D interrupt to flag the end of conversion.

The routine ReadAD will start a read of the A/D channel identified by R7, and wait for the conversion to complete, polling the A/D interrupt flag. The result is returned in the accumulator.

```
ReadAD: MOV  A,#08h          ;Basic A/D
                             ;start
                             ;command.
        ORL  A,R7            ;Add channel
                             ;# to be read.
        MOV  ADCON,A;        ;Start A/D.
ADLoop: MOV  A,ADCON         ;Get A/D
                             ;status.
        JNB  ACC.4,ADLoop;Wait for
                             ;ADCI
                             ;(A/D
                             ;finished).
        MOV  A,ADAT          ;Get
                             ;conversion
                             ;result
        MOV  ADCON,#0        ;Clear ADCI.
        RET
```

The routine StartAD will start a read of the A/D channel identified by R7 and exit back to the calling program. When the conversion is complete, the A/D interrupt occurs, calling the A/D interrupt service routine. The result of the conversion is returned in register R6.

```
StartAD: MOV  A,#08h         ;Basic A/D
                             ;start
                             ;command.
         ORL  A,R7           ;Add channel
                             ;# to be read.
         MOV  ADCON,A        ;Start A/D.
         RET
         .
         .
         .
         ORG  2Bh            ;A/D interrupt
                             ;address.
ADInt:   MOV  R6,ADAT        ;Get
                             ;conversion
                             ;result.
         MOV  ADCON,#0       ;Clear ADCI.
         RETI
```

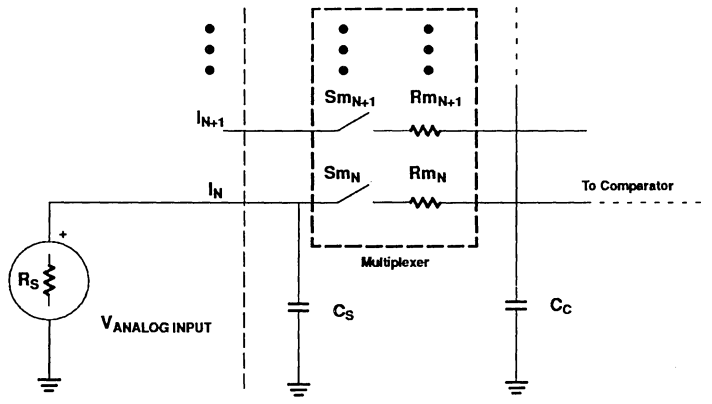
Watchdog Timer

The purpose of the watchdog timer is to reset the microcontroller within a reasonable amount of time if it enters an erroneous state, possibly due to a programming error, electrical noise, or RFI. When enabled, the watchdog circuit will generate a system reset if the user program fails to "feed" (or reload) the watchdog within a predetermined amount of time.

The watchdog timer implemented on the 8XC550 has a programmable interval and can thus be fine tuned to a particular application. If the watchdog function is not used, the timer may still be used as a versatile general purpose timer.

The watchdog function consists of a programmable 13-bit prescaler, and an 8-bit main timer. The main timer is clocked by a tap taken from one of the top 8 bits of the prescaler. The prescaler is incremented once every machine cycle, or 1/12 of the oscillator frequency. Thus, the main counter can be clocked as often as once every 64 machine cycles or as seldom as once every 8192 machine cycles.

When clocked, the main counter decrements. If the main watchdog counter reaches zero, a system reset will occur. To prevent the watchdog timer from under-flowing, the watchdog must be fed before it counts down to zero. When the watchdog is fed, the contents of the WDL register are loaded into the main watchdog counter and the prescaler is cleared.



Rm = 0.5 - 3 kohms
CS + CC = 15pF maximum
RS = Recommended < 9.6 kohms for 1 LSB @ 12MHz

NOTE:
Because the analog to digital converter has a sampled-data comparator, the input looks capacitive to a source. When a conversion is initiated, switch Sm closes for 8tcy (8µs @ 12MHz crystal frequency) during which time capacitance Cs + Cc is charged. It should be noted that the sampling causes the analog input to present a varying load to an analog source.

Figure 1. A/D Input: Equivalent Circuit

WDCON Register

MSB							LSB
PRE2	PRE1	PRE0	X	X	WDRUN	WDTOF	WDMOD

Symbol	Position	Function
WDCON.7	PRE2	Prescaler select (read/write).
WDCON.6	PRE1	These bits select the
WDCON.5	PRE0	prescaler divide ratio according to the following table:

PRE2	PRE1	PRE0	DIVISOR (FROM f_{OSC})
0	0	0	12 × 64
0	0	1	12 × 64 × 2
0	1	0	12 × 64 × 4
0	1	1	12 × 64 × 8
1	0	0	12 × 64 × 16
1	0	1	12 × 64 × 32
1	1	0	12 × 64 × 64
1	1	1	12 × 64 × 128

WDCON.4 – Not used
WDCON.3 – Not used
WDCON.2 WDRUN Run control (read/write). This bit turns the timer on (WDRUN = 1) or off (WDRUN = 0) if the timer mode has been selected.
WDCON.1 WDTOF Timeout flag (read /write). This bit is set when the watchdog timer underflows. It is cleared by an external reset and can be cleared by software.
WDCON.0 WDMOD Mode selection (read/write). When WDMOD = 1, the watchdog is selected; when WDMOD = 0, the timer is selected. Selecting the watchdog mode automatically disables power-down mode. WDMOD is cleared by external reset. Once the watchdog mode is selected, this bit can only be cleared by writing a 0 to this bit and then performing a feed operation.

A very specific sequence of events must take place to feed the watchdog timer; it cannot be fed accidentally by a runaway program. The following routines demonstrate setting up and feeding the watchdog timer. These routines apply to all versions of the 8XC550 except the ROM part when running from internal program memory.

This routine sets up and starts the watchdog timer. This is not necessary for internal ROM operation, because setup of the watchdog timer on masked ROM parts is accomplished directly via ROM mask options.

```
SetWD:  MOV   WDL,#0FFh     ;Set watch-
                            ;dog
                            ;reload value.
        MOV   WDCON,#0E5    ;Set up timer
                            ;prescaler,
                            ;mode, and
                            ;run bits.
        ACALL FeedWD        ;Start watch-
                            ;dog with a
                            ;feed
                            ;operation.
        RET
```

This routine executes a watchdog timer feed operation, causing the timer to reload from WDL. Interrupts must be disabled during this operation due to the fact that the two feed registers must be loaded on consecutive instruction cycles, or a system reset will occur immediately.

```
FeedWD:CLR  EA          ;This
                        ;sequence
                        ;must not be
                        ;interrupted.
       MOV WFEED1,#0A5h;First
                        ;instruction
                        ;of feed
                        ;sequence.
       MOV WFEED2,#05Ah;Second
                        ;instruction
                        ;of feed
                        ;sequence.
       SETB EA          ;Turn
                        ;interrupts
                        ;back on.
       RET
```

An interrupt is available to allow the watchdog timer to be used as a general purpose timer in applications where the watchdog function is not needed. The timer operates in the same manner when used as a general purpose timer except that the timer interrupt is generated on timer underflow instead of a chip reset. Refer to the 87C550 data sheet for additional information on watchdog timer operation.

Interrupts

The 8XC550 interrupt structure is a seven-source, two-priority level interrupt system similar to that of the standard 80C51 microcontroller. The interrupt sources are listed below in the order of their internal polling sequence. This is the order in which simultaneous interrupts of the same priority level would be serviced.

Interrupt Priorities

PRIORITY	SOURCE	VECTOR ADDRESS	FUNCTION
Highest	INT0	0003H	External interrupt 0
	TF0	000BH	Counter/ timer 0 overflow
	INT1	0013H	External interrupt 1
	TF1	001BH	Counter/ timer 1 overflow
	TI & RI	0023H	Serial port transmit/ receive
	ADCI	002BH	A/D converter conversion complete
Lowest	WDTOF	0033H	Watchdog timer overflow (only when not in watchdog mode)

Interrupt Control Registers

The standard 80C51 interrupt enable and priority registers have been modified slightly to take into account the additional interrupt sources of the 8XC550.

Interrupt Enable Register

MSB							LSB
EA	EWD	EAD	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
EA	IE.7	Global interrupt enable
EWD	IE.6	Watchdog timer overflow
EAD	IE.5	A/D conversion complete
ES	IE.4	Serial port transmit or receive
ET1	IE.3	Timer 1 overflow
EX1	IE.2	External interrupt 1
ET0	IE.1	Timer 0 overflow
EX0	IE.0	External interrupt 0

Interrupt Priority Register

MSB							LSB
–	PWD	PAD	PS	PT1	PT1	PX0	PX0

Symbol	Position	Function
PWD	IP.6	Watchdog timer
PAD	IP.5	A/D conversion
PS	IP.4	Serial port interrupt
PT1	IP.3	Timer 1 interrupt
PX1	IP.2	External interrupt 1
PT0	IP.1	Timer 0 interrupt
PX0	IP.0	External interrupt 0

Power-Down and Idle Modes

The 8XC550 includes the standard 80C51 power-down and idle modes of reduced power consumption. In addition, the 8XC550 includes an option to separately turn off the serial port for extra power savings when it is not needed. Also, the individual functional blocks such as the counter/timers are automatically disabled when they are not running. This actually turns off the clocks to the block in question, resulting in additional power savings. Note that when the watchdog timer is operating, the processor is inhibited from entering the power-down mode. This is due to the fact that the oscillator is stopped in the power-down mode, which would effectively turn off the watchdog timer. In keeping with the purpose of the watchdog timer, the processor is prevented from accidentally entering power-down due to some erroneous operation.

Power Control Register

MSB							LSB
SMOD	SIDL	–	–	GF1	GF0	PD	IDL

Symbol	Position	Function
SMOD	PCON.7	Double baud rate bit. When set to a 1 and Timer 1 is used to generate baud rate, and the serial port is used in modes 1, 2, or 3.

Symbol	Position	Function
SIDL	PCON.6	Separately idles the serial port for additionalpower savings.
–	PCON.5	Reserved
–	PCON.4	Reserved
GF1	PCON.3	General-purpose flag bit.
GF0	PCON.2	General-purpose flag bit.
PD	PCON.1	Power-down bit. Starting this bit activates power-down operation.
IDL	PCON.0	Idle mode bit. Setting this bit activates idle mode operation.

If 1s are written to PD and IDL at the same time, PD takes precedence.

CMOS single-chip 8-bit microcontroller with A/D and watchdog timer

80C550/83C550/87C550

DESCRIPTION

The Philips 8XC550 is a high-performance microcontroller fabricated with Philips high-density CMOS technology. This Philips CMOS technology combines the high speed and density characteristics of HMOS with the low power attributes of CMOS. Philips epitaxial substrate minimizes latch-up sensitivity. The CMOS 8XC550 has the same instruction set as the 80C51.

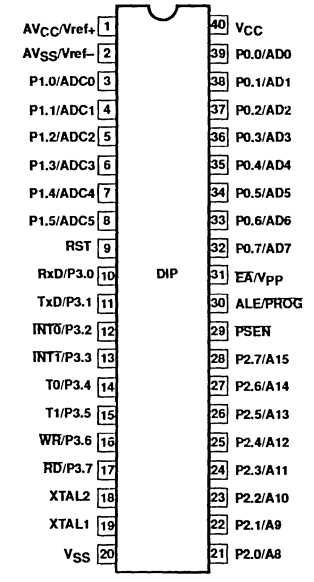
The 8XC550 contains a 4k × 8 EPROM (87C550)/ROM (83C550)/ROMless (80C550 has no program memory on-chip), a 128 × 8 RAM, 8 channels of 8-bit A/D, four 8-bit ports (port 1 is input only), a watchdog timer, two 16-bit counter/timers, a seven-source, two-priority level nested interrupt structure, a serial I/O port for either multi-processor communications, I/O expansion or full duplex UART, and an on-chip oscillator and clock circuits.

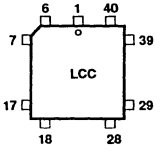
In addition, the 8XC550 has two software selectable modes of power reduction — idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

FEATURES

- 80C51 based architecture
  - 4k × 8 EPROM (87C550)/ ROM (83C550)
  - 128 × 8 RAM
  - Eight channels of 8-bit A/D
  - Two 16-bit counter/timers
  - Watchdog timer
  - Full duplex serial channel
  - Boolean processor
- Memory addressing capability
  - 64k ROM and 64k RAM
- Power control modes:
  - Idle mode
  - Power-down mode
- CMOS and TTL compatible
- One speed range at V_{CC} = 5V ±10%
  - 3.5 to 16MHz
- Four package styles
- Extended temperature ranges
- OTP package available

PIN CONFIGURATIONS



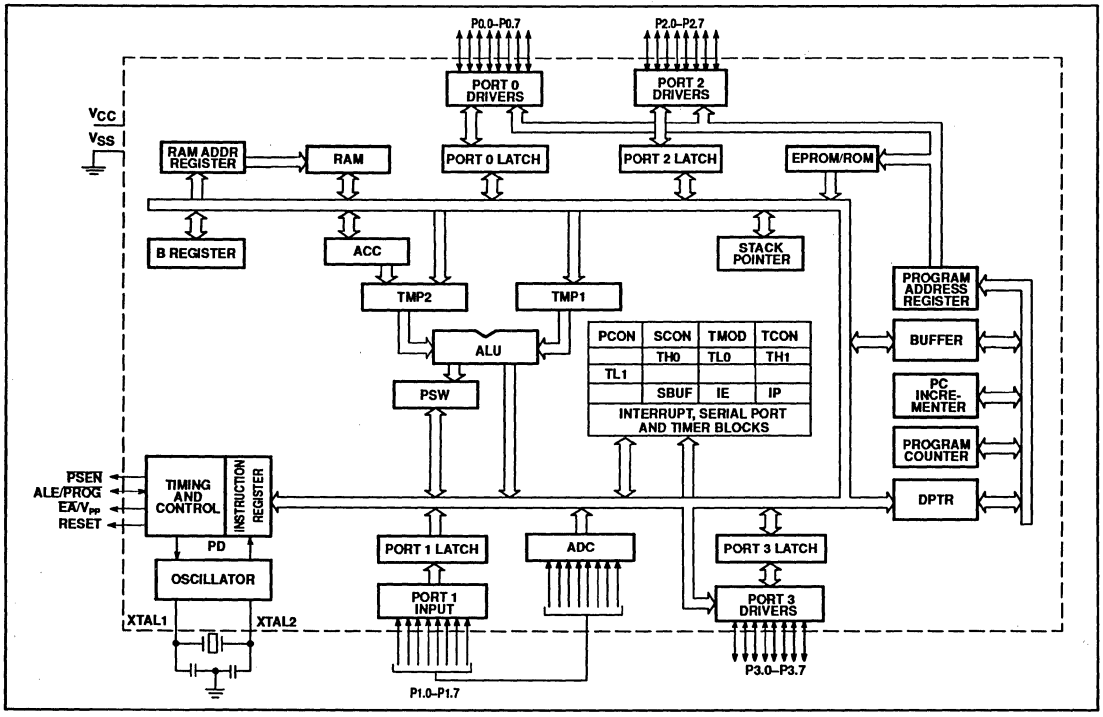


Pin	Function	Pin	Function	Pin	Function
1	AV_{CC}	16	P3.2/INT0	31	P2.6/A14
2	Vref+	17	P3.3/INT1	32	P2.7/A15
3	Vref–	18	P3.4/T0	33	PSEN
4	AV_{SS}	19	P3.5/T1	34	ALE/PROG
5	P1.0/ADC0	20	P3.6/WR	35	EA/V_{PP}
6	P1.1/ADC1	21	P3.7/RD	36	P0.7/AD7
7	P1.2/ADC2	22	XTAL2	37	P0.6/AD6
8	P1.3/ADC3	23	XTAL1	38	P0.5/AD5
9	P1.4/ADC4	24	V_{SS}	39	P0.4/AD4
10	P1.5/ADC5	25	P2.0/A8	40	P0.3/AD3
11	P1.6/ADC6	26	P2.1/A9	41	P0.2/AD2
12	P1.7/ADC7	27	P2.2/A10	42	P0.1/AD1
13	RST	28	P2.3/A11	43	P0.0/AD0
14	P3.0/RxD	29	P2.4/A12	44	V_{CC}
15	P3.1/TxD	30	P2.5/A13		

PART NUMBER SELECTION

ROMless	ROM	EPROM	FREQUENCY	TEMPERATURE AND PACKAGE
		P87C550EBF FA	3.5 to 16MHz	0 to +70°C, ceramic DIP
		P87C550EBL KA	3.5 to 16MHz	0 to +70°C, ceramic LCC
P80C550EBP N	P83C550EBP N	P87C550EBP N	3.5 to 16MHz	0 to +70°C, plastic DIP
P80C550EBA A	P83C550EBA A	P87C550EBA A	3.5 to 16MHz	0 to +70°C, plastic LCC
P80C550EFP N	P83C550EFP N	P87C550EFP N	3.5 to 16MHz	–40 to +85°C, plastic DIP
P80C550EFA A	P83C550EFA A	P87C550EFA A	3.5 to 16MHz	–40 to +85°C, plastic LCC
		P87C550BFL KA	3.5 to 16MHz	–40 to +85°C, ceramic LCC
		P87C550EFL KA	3.5 to 16MHz	–40 to +85°C, ceramic DIP

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	20	24	I	**Ground:** 0V reference.
V_{CC}	40	44	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
AV_{CC}	1	1	I	**Analog Power Supply:** Analog supply voltage.
AV_{SS}	2	4	I	**Analog Ground:** Analog 0V reference.
Vref+		2	I	**Vref:** A/D converter reference level inputs. Note that these references are combined with AV_{CC} and AV_{SS} in the 40-pin DIP package.
Vref–		3	I	
P0.0–0.7	39–32	43–36	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 also outputs the code bytes during program verification in the S87C550. External pull-ups are required during program verification.
P1.0–P1.7	3–8	5–12	I	**Port 1:** Port 1 is an 8-bit input only port (6-bit in the DIP package; bits P1.6 and P1.7 are not implemented). Port 1 digital input can be read out any time.
ADC0–ADC7	3–8	5–12		**ADCx:** Inputs to the analog multiplexer input of the 8-bit A/D. There are only six A/D inputs in the DIP package.
P2.0–P2.7	21–28	25–32	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	14–21	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the SC80C51 family, as listed below:
	10	14	I	**RxD (P3.0):** Serial input port
	11	15	O	**TxD (P3.1):** Serial output port
	12	16	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	17	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	18	I	**T0 (P3.4):** Timer 0 external input
	15	19	I	**T1 (P3.5):** Timer 1 external input
	16	20	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	21	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	13	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE/$\overline{PROG}$	30	34	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{PSEN}$	29	33	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$/V_{PP}	31	35	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 0FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 0FFFH. For the 80C550 ROMless part, $\overline{EA}$ must be held low for the part to operate properly. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	19	23	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	22	O	**Crystal 2:** Output from the inverting oscillator amplifier.

Table 1. Interrupt Priorities

PRIORITY	SOURCE	VECTOR ADDRESS	FUNCTION
Highest	$\overline{INT0}$	0003H	External interrupt 0
	T0	000BH	Timer flag 0
	$\overline{INT1}$	0013H	External interrupt 1
	T1	001BH	Timer flag 1
	SIO	0023H	Serial port interrupt
	ADC	002BH	A/D conversion complete
Lowest	WD	0033H	Watchdog timer

INTERRUPTS

The interrupt structure is a seven source, two level interrupt system similar to that of the 80C51. The interrupt sources are listed in Table 1 in order of polling sequence priority (highest to lowest). Note that the watchdog timer function is available only if the watchdog timer function is disabled and the watchdog timer is used as a general purpose timer.

Interrupt Control Registers

The interrupt enable and the interrupt priority registers have been modified to take into account the different interrupt sources of the 8XC550. In all other respects, their operation is identical to that of the 80C51. Setting a bit in the IE register enables the interrupt; clearing the bit disables the interrupt. All bits are cleared by reset. See Figure 1 for interrupt register formats.

SERIAL COMMUNICATIONS

The serial port operation is identical to that of the 80C51. In order to conserve power, another bit (SIDL) has been added to the PCON register that idles the serial port when it is not being used. This bit is cleared by reset. See Figure 2.

A/D CONVERTER

The analog input circuitry consists of an 8-input analog multiplexer and an analog-to-digital converter with 8-bit resolution. In the LCC package, the analog reference voltage and analog power supplies are connected via separate input pins; in the DIP package, Vref+ is combined with AV_{CC} and Vref– is combined with AV_{SS}. The analog inputs are alternate functions to port 1, which is an input only port. Digital input to port 1 can be read any time during an A/D conversion. Care should be exercised in mixing analog and digital signals on port 1, because cross talk from the digital input signals can degrade the A/D conversion accuracy of the analog input. An A/D conversion requires 40 machine cycles.

The A/D converter is controlled by the ADCON special function register. The input channel to be converted is selected by the analog multiplexer by setting ADCON register bits, ADDR2–ADDR0 (see Figure 3). These bits can only be changed when ADCI and ADCS are both low.

The completion of the 8-bit ADC conversion is flagged by ADCI in the ADCON register and the result is stored in the special function register ADAT. The functions of the ADCI and ADCS are:

ADCI	ADCS	Operation
0	0	ADC not busy. A conversion can be started.
0	1	ADC busy. Start of a new conversion is blocked.
1	0	Conversion completed. start of new conversion is blocked.
1	1	Not possible.

An ADC conversion in progress is unaffected by a software ADC start. The result of a completed conversion remains unaffected provided ADCI remains at a logic 1. While ADCS is a logic 1 or ADCI is a logic 1, a new ADC START will be blocked and consequently lost. An A/D conversion in progress will be aborted when the idle or power-down mode is entered. The result of a completed conversion (ADCI = logic 1) remains unaffected when entering the idle mode, but will be lost if power-down mode is entered.

MSB INTERRUPT ENABLE REGISTER (IE) LSB

EA	EWD	EAD	ES	ET1	EX1	ET0	EX0

BIT	SYMBOL	FUNCTION
IE.7	EA	GLOBAL INTERRUPT DISABLE WHEN EA = 0
IE.6	EWD	WATCHDOG TIMER OVERFLOW
IE.5	EAD	A/D CONVERSION COMPLETE
IE.4	ES	SIO INTERRUPT
IE.3	ET1	TIER 1 OVERFLOW
IE.2	EX1	EXTERNAL INTERRUPT 1
IE.1	ET0	TIMER 0 OVERFLOW
IE.0	EX0	EXTERNAL INTERRUPT 0

MSB INTERRUPT PRIORITY REGISTER (IP) LSB

X	PWD	PAD	PS	PT1	PX1	PT0	PX0

BIT	SYMBOL	FUNCTION
IP.6	PWD	WATCHDOG TIMER
IP.5	PAD	A/D CONVERTER
IP.4	PS	SIO INTERRUPT
IP.3	PT1	T1 INTERRUPT
IP.2	PX1	$\overline{INT1}$ INTERRUPT
IP.1	PT0	T0 INTERRUPT
IP.0	PX0	$\overline{INT0}$ INTERRUPT

NOTE:
The interrupt priority register function is identical to that of the 80C51. A '1' sets that interrupt to the high priority level and a '0' sets the interrupt to the low priority level.

Figure 1. Interrupt Registers

MSB LSB

SMOD	SIDL	X	X	GF1	GF0	PD	IDL

BIT	SYMBOL	FUNCTION
PCON.7	SMOD	DOUBLE BAUD RATE
PCON.7	SIDL	SERIAL PORT IDLE
PCON.5	X	RESERVED FOR FUTURE USE
PCON.4	X	RESERVED FOR FUTURE USE
PCON.3	GF1	GENERAL PURPOSE FLAG BIT
PCON.2	GF0	GENERAL PURPOSE FLAG BIT
PCON.1	PD	POWER DOWN BIT
PCON.0	IDL	IDLE MODE BIT

NOTE:
The PCON register is at SFR byte address 87H. Its contents following a reset are 00XX0000.

Figure 2. Power Control Register (PCON)

MSB LSB

X	X	X	ADCI	ADCS	AADR2	AADR1	AADR0

INPUT CHANNEL SELECTION

ADDR2	ADDR1	ADDR0	INPUT PIN
0	0	0	ADC0
0	0	1	ADC1
0	1	0	ADC2
0	1	1	ADC3
1	0	0	ADC4
1	0	1	ADC5
1	1	0	ADC6
1	1	1	ADC7

BIT	SYMBOL	FUNCTION
ADCON.7	–	NOT USED
ADCON.6	–	NOT USED
ADCON.5	–	NOT USED
ADCON.4	ADCI	ADC INTERRUPT FLAG. This flag is set when an ADC conversion result is ready to be read. An interrupt is invoked if the A/D interrupt is enabled. The flag must be cleared by software. It cannot be set by software.
ADCON.3	ADCS	ADC START AND STATUS. Setting this flag starts an A/D conversion. The ADC logic insures that this signal is high while the ADC is busy. On completion of the conversion, ADCS is reset at the same time the interrupt flag ADCI is set. ADCS cannot be reset by software.
ADCON.2	ADDR2	ANALOG INPUT SELECT 2
ADCON.1	ADDR1	ANALOG INPUT SELECT 1
ADCON.0	ADDR0	ANALOG INPUT SELECT 0

Figure 3. A/D Control Register (ADCON)

WATCHDOG TIMER

The watchdog timer is not directly loadable by the user. Instead, the value to be loaded into the main timer is held in an autoload register or is part of the mask ROM programming. In order to cause the main timer to be loaded with the appropriate value, a special sequence of software action must take place. This operation is referred to as feeding the watchdog timer.

To feed the watchdog, two instructions must be sequentially executed successfully. No intervening instruction fetches are allowed, so interrupts should be disabled before feeding the watchdog. The instructions should move A5H to the WFEED1 register and then 5AH to the WFEED2 register. If WFEED1 is correctly loaded and WFEED2 is not correctly loaded, then an immediate underflow will occur.

The watchdog timer subsystem has two modes of operation. Its principal function is a watchdog timer. In this mode it protects the system from incorrect code execution by causing a system reset when the watchdog timer underflows as a result of a failure of software to feed the timer prior to the timer reaching its terminal count. If the user does not employ the watchdog function, the watchdog subsystem can be used as a timer. In this mode, reaching the terminal count sets a flag which can be used to generate an interrupt. In most other respects, the timer mode possesses the characteristics of the watchdog mode. This is done to protect the integrity of the watchdog function.

The watchdog timer subsystem consists of a prescaler and a main counter. The prescaler has 8 selectable taps off the final stages and the output of a selected tap provides the clock to the main counter. The main counter is the section that is loaded as a result of the software feeding the watchdog and it is the section that causes the system reset (watchdog mode) or time-out flag to be set (timer mode) if allowed to reach its terminal count.

Programming the Watchdog Timer

Both the EPROM and ROM devices have a set of SFRs for holding the watchdog autoload values and the control bits. The watchdog time-out flag is present in the watchdog control register and operates the same in all versions. In the EPROM device, the watchdog parameters (autoload value and control) are always taken from the SFRs. In the ROM device, the watchdog parameters can be mask programmed or taken from the SFRs. The selection to take the watchdog parameters from the SFRs or from the mask programmed values is controlled by EA (external access). When EA is high (internal ROM access), the watchdog parameters are taken from the mask programmed values. If the watchdog is masked programmed to the timer mode, then the autoload values and the pre-scaler taps are taken from the SFRs. When EA is low (external access), the watchdog parameters are taken from the SFRs. The user should be able to leave code in his program which initializes the watchdog SFRs even though he has migrated to the mask ROM part. This allows no code changes from EPROM prototyping to ROM coded production parts.

Watchdog Detailed Operation

EPROM Device (and ROMless Operation: EA = 0)

In the ROMless operation (ROM part, EA = 0) and in the EPROM device, the watchdog operates in the following manner.

Whether the watchdog is in the watchdog or timer mode, when external RESET is applied, the following takes place:

- Watchdog mode bit set to timer mode.
- Watchdog run control bit set to OFF.
- Autoload register set to FF (max count).
- Watchdog time-out flag cleared.
- Prescaler is cleared.
- Prescaler tap set to the highest divide.
- Autoload takes place.

The watchdog can be fed even though it is in the timer mode.

Note that the operational concept is for the watchdog mode of operation, when coming out of a hardware reset, the software should load the autoload registers, set the mode to watchdog, and then feed the watchdog (cause an autoload). The watchdog will now be starting at a known point.

If the watchdog is in the watchdog mode and running and happens to underflow at the time the external RESET is applied, the watchdog time-out flag will be cleared.

When the watchdog is in the watchdog mode and the watchdog underflows, the following action takes place:

- Autoload takes place.
- Watchdog time-out flag is set
- Timer mode interrupt flag unchanged.
- Mode bit unchanged.
- Watchdog run bit unchanged.
- Autoload register unchanged.
- Prescaler tap unchanged.
- All other device action same as external reset.

Note that if the watchdog underflows, the program counter will start from 00H as in the case of an external reset. The watchdog time-out flag can be examined to determine if the watchdog has caused the reset condition. The watchdog time-out flag bit can be cleared by software.

When the watchdog is in the timer mode and the timer software underflows, the following action takes place:

- Autoload takes place.
- Watchdog time-out flag is set
- Mode bit unchanged.
- Watchdog run bit unchanged.
- Autoload register unchanged.
- Prescaler tap unchanged.

The timer mode interrupt flag is cleared when the interrupt routine is invoked. This bit can also be cleared directly by software without a software feed operation.

Mask ROM Device (EA = 1)
In the mask ROM device, the watchdog mode bit (WDMOD) is mask programmed and the bit in the watchdog command register is read only and reflects the mask programmed selection. If the mask programmed mode bit selects the timer mode, then the watchdog run bit (WDRUN) operates as described under EPROM Device. If the mask programmed bit selects the watchdog mode, then the watchdog run bit has no effect on the timer operation.

Watchdog Function
The watchdog consists of a programmable prescaler and the main timer. The prescaler derives its clock from the on-chip oscillator. The prescaler consists of a divide by 12 followed by a 13 stage counter with taps from stage 6 through stage 13. The tap selection is programmable. The watchdog main counter is a down counter clocked (decremented) each time the programmable prescaler underflows. The watchdog generates an underflow signal (and is autoloaded) when the watchdog is at count 0 and the clock to decrement the watchdog occurs. The watchdog is 8 bits long and the autoload value can range from 0 to FFH. (The autoload value of 0 is permissible since the prescaler is cleared upon autoload).

This leads to the following user design equations. Definitions :t_{OSC} is the oscillator period, N is the selected prescaler tap value, W is the main counter autoload value, t_{MIN} is the minimum watchdog time-out value (when the autoload value is 0), t_{MAX} is the maximum time-out value (when the autoload value is FFH), t_D is the design time-out value.

$$t_{MIN} = t_{OSC} \times 12 \times 64$$

$$t_{MAX} = t_{MIN} \times 128 \times 256$$

$$t_D = t_{MIN} \times 2^{PRESCALER} \times W$$
(where prescaler = 0, 1, 2, 3, 4, 5, 6, or 7)

Note that the design procedure is anticipated to be as follows. A t_{MAX} will be chosen either from equipment or operation considerations and will most likely be the next convenient value higher than t_D. (If the watchdog were inadvertently to start from FFH, an overflow would be guaranteed, barring other anomalies, to occur within t_{MAX}). Then the value for the prescaler would be chosen from:

$$\text{prescaler} = \log2\,(t_{MAX} / (t_{OSC} \times 12 \times 256)) - 6$$

This then also fixes t_{MIN}. An autoload value would then be chosen from:

$$W = t_D / t_{MIN} - 1$$

The software must be written so that a feed operation takes place every t_D seconds from the last feed operation. Some tradeoffs may need to be made. It is not advisable to include feed operations in minor loops or in subroutines unless the feed operation is a specific subroutine.

Watchdog Control Register (WDCON) (Bit Addressable) Address C0
The following bits of this register are read only in the ROM part when EA is high: WDMOD, PRE0, PRE1, and PRE2. That is, the register will reflect the mask programmed values. In the ROM part with EA high, these bits are taken from mask coded bits and are not readable by the program. WDRUN is read only in the ROM part when EA is high and WDMOD is in the watchdog mode. When WDMOD is in the timer mode, WDRUN functions normally (see Figure 4).

The parameters written into WDMOD, PRE0, PRE1, and PRE2 by the program are not applied directly to the watchdog timer subsystem. The watchdog timer subsystem is directly controlled by a second register which stores these bits. The transfer of these bits from the user register (WDMOD) to the second control register takes place when the watchdog is fed. This prevents random code execution from directly foiling the watchdog function. This does not affect the operation where these bits are taken from mask coded values.

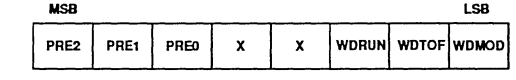
OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the Block Diagram, page 346).

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. the instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. An A/D conversion in progress will be aborted when idle mode is entered. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.



BIT	SYMBOL	FUNCTION
WDCON.7	PRE2	PRESCALER SELECT (READ/WRITE)
WDCON.6	PRE1	PRESCALER SELECT (READ/WRITE)
WDCON.5	PRE0	PRESCALER SELECT (READ/WRITE). These bits select the prescaler divide ratio according to the following table:

PRE2	PRE1	PRE0	DIVISOR (FROM t_{osc})
0	0	0	12 X 64
0	0	1	12 X 64 X 2
0	1	0	12 X 64 X 4
0	1	1	12 X 64 X 8
1	0	0	12 X 64 X 16
1	0	1	12 X 64 X 32
1	1	0	12 X 64 X 64
1	1	1	12 X 64 X 128

WDCON.4	–	NOT USED
WDCON.3	–	NOT USED
WDCON.2	WDRUN	RUN CONTROL (READ/WRITE). This bit turns the timer on (WDRUN = 1) or off (WDRUN = 0) if the timer mode has been selected.
WDCON.1	WDTOF	TIMEOUT FLAG (READ/WRITE). This bit is set when the watchdog timer underflows. It is cleared by an external reset and can be cleared by software.
WDCON.0	WDMOD	MODE SELECTION (READ/WRITE). When WDMOD = 1, the watchdog mode is selected; when WDMOD = 0, the timer mode is selected. Selecting the watchdog mode automatically disables power-down mode. WDMOD is cleared by external reset. Once the watchdog mode is selected, this bit can only be cleared by writing a 0 to this bit and then performing a feed operation.

Figure 4. Watchdog Control Register (WDCON)

Programmable Idle Modes

The programmable idle modes have been dispersed throughout the functional blocks. Each block has its own ability to be disabled. For example, if timer 0 is not commanded to be running (TR = 0), then the clock to the timer is disabled resulting in an idle mode power saving. An additional idle control bit has been added to the serial communications port.

A/D Operation in Idle Mode

When in the idle mode, the A/D converter will be disabled. However, the current through the Vref pins will be present and will not be reduced internally in either the idle or the power-down modes. It is the responsibility of the user to disconnect Vref to reduce power supply current.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. The power-down mode can be terminated by a Reset in the same way as in the 80C51. The control bits for the reduced power modes are in the special function register PCON.

DESIGN CONSIDERATIONS

At power-on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

When the idle mode is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write when idle is terminated by reset, the instruction following the one that invokes idle should not be one that writes to a port pin or to external memory. Table 2 shows the state of I/O ports during low current operating modes.

Table 2. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Encryption Table

The encryption table is a feature of the 83C550 and 87C550 that protects the code from being easily read by anyone other than the programmer. The encryption table is 32 bytes of code that are exclusive NORed with the program code data as it is read out. The first byte is XNORed with the first location read, the second with the second read, etc.

After the encryption table has been programmed, the user has to know its contents in order to correctly decode the program code data. The encryption table itself cannot be read out.

For the EPROM (87C550) part, the encryption table is programmed in the same manner as the program memory, but using the "Pgm Encryption Table" levels specified in Table 4. After the encryption table is programmed, verification cycles will produce only encrypted information.

For the ROM part (83C550) the encryption table information is submitted with the ROM code as shown in Table 3.

Lock Bits

There are two lock bits on the 83C550 and 87C550 that, when set, prevent the program data memory from being read out or programmed further.

After the first lock bit is programmer, the external MOVC instruction is disabled, and for the 87C550, further programming of the code memory or the encryption table is disabled. The other lock bit can of course still be programmed. With only lock bit one programmed, the memory can still be read out for program verification. After the second lock bit is programmed, it is no longer possible to read out (verify) the program memory.

To program the lock bits for the 87C550, repeat the programming sequence using the "Pgm Lock Bit" levels specified in Table 4. For the masked ROM 83C550 the lock bit information is submitted with the ROM code as shown in Table 3.

ROM Code Submission

When submitting a ROM code for the 83C550, the following must be specified:

1. The 4k byte user ROM program.
2. The 32 byte ROM encryption key.
3. The ROM lock bits.
4. The watchdog timer parameters.

This information can be submitted in an EPROM (2764) or hex file with the format specified in Table 3.

Table 3. ROM Code Submittal Requirements

ADDRESS	CONTENT	BIT(s)	COMMENT
0000H to 0FFFH	Data	7:0	User ROM data
1000H to 101FH	Key	7:0	ROM encryption key; FFH = no encryption
1020H	Lock	0	ROM lock bit 1
1020H	Lock	1	ROM lock bit 2 0 = enable security feature 1 = disable security feature
1030H	WMOD	0	Watchdog mode bit; 00H = timer mode 01H = watchdog mode
1031H	PRE2:0	2:0	Watchdog prescaler selection; 00H = divide by 12 x 64 07H = divide by 12 x 64 x 128 (see specification)
1032H	WD	7:0	Watchdog autoload value (see specification)

Electrical Deviations from Commercial Specifications for Extended Temperature Range
DC and AC parameters not included here are the same as in the commercial temperature range table.

DC ELECTRICAL CHARACTERISTICS

T_{amb} = –40°C to +85°C, V_{CC} = 5V ±10% (87C550), V_{CC} = 5V ±20% (80/83C550), V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNIT
			MIN	MAX	
V_{IL}	Input low voltage, except $\overline{EA}$		–0.5	$0.2V_{CC}$–0.15	V
V_{IL1}	Input low voltage to $\overline{EA}$		0	$0.2V_{CC}$–0.35	V
V_{IH}	Input high voltage, except XTAL1, RST		$0.2V_{CC}$+1	V_{CC}+0.5	V
V_{IH1}	Input high voltage to XTAL1, RST		$0.7V_{CC}$+0.1	V_{CC}+0.5	V
I_{IL}	Logical 0 input current, ports 2, 3	V_{IN} = 0.45V		–75	µA
I_{TL}	Logical 1-to-0 transition current, ports 2, 3	V_{IN} = 2.0V		–750	µA
I_{CC}	Power supply current: Active mode Idle mode Power down mode	V_{CC} = 4.5–5.5V, Frequency range = 3.5 to 16MHz		 35 6 50	 mA mA µA

ADC DC ELECTRCIAL CHARACTERISTICS

AV_{CC} = 5V ±10%, AV_{SS} = 0V, T_{amb} = –40°C to 85°C, unless otherwise specified

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNIT
			MIN	MAX	
AV_{CC}	Analog supply	$AV_{CC} = V_{CC} \pm 0.2$	4.5	5.5	V
Vref	Analog reference; AVref+ AVref-		AV_{SS} – 0.2	AV_{CC} + 0.2	V
AI_{CC}	Analog operating supply current	See note 1		3.0	mA
AV_{IN}	Analog input voltage		AV_{SS} – 0.2	AV_{CC} + 0.2	V
A_{IC}, C_{IA}	Analog input capacitance			15	pF
t_{ADS}	Sampling time			$8t_{CY}$	
t_{ADC}	Conversion time			$40t_{CY}$	
Ae	Absolute voltage error			±1.5	LSB
E_{RA}	Relative accuracy			±1	LSB
OSe	Offset error	See note 1		±10	mV
Ge	Gain error	See note 1		0.4	%
M_{CTC}	Channel-to-channel matching			±1	LSB
Ct	Crosstalk	0 – 100kHz		–60	dB
Rref	Resistance between AVref+ and AVref–		1.0	10.0	KΩ
AI_{ID}	Idle mode supply current	See note 4		50	µA
AI_{PD}	Power down supply current	See note 4		50	µA

NOTES:
1. Conditions: Vref+ = 4.99712V, Vref– = 0V. AI_{CC} value does not include the resistor ladder current. For the 40-pin package, where the Vref– inputs are connected to AV_{CC} and AV_{SS}, the current AI_{CC} will be increased by the register ladder current and may exceed the maximum shown here.
2. The resistor ladder network is not disconnected in the power-down or idle modes. Thus to conserve power, the user must remove AV_{CC} and Vref+.
3. If the A/D function is not required, or if the A/D function is only needed periodically, AV_{CC} can be removed without affecting the operation of the digital circuitry. Contents of ADCON and ADAT are not guaranteed to be valid. Digital inputs P1.0 to P1.7 will not function normally. No digital outputs are present on these pins.
4. For this test, the Analog inputs must be at the supplies (either V_{DD} or V_{SS}).

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Operating temperature under bias	–40 to +85	°C
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ pin to V_{SS} (87C550 only)	0 to +13.0	V
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Input, output current on any two I/O pins	±10	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.5	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10% (87C550), V_{CC} = 5V ±20% (80/83C550), V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			MIN	TYPICAL[1]	MAX	
V_{IL}	Input low voltage, except $\overline{EA}$[7]		–0.5		$0.2V_{CC}$–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$[7]		0		$0.2V_{CC}$–0.3	V
V_{IH}	Input high voltage, except XTAL1, RST[7]		$0.2V_{CC}$+0.9		V_{CC}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST[7]		$0.7V_{CC}$		V_{CC}+0.5	V
V_{OL}	Output low voltage, ports 2, 3	I_{OL} = 1.6mA[2]			0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$	I_{OL} = 3.2mA[2]			0.45	V
V_{OH}	Output high voltage, ports 2, 3, ALE, $\overline{PSEN}$[3]	I_{OH} = –60µA, I_{OH} = –25µA I_{OH} = –10µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
V_{OH1}	Output high voltage (port 0 in external bus mode)	I_{OH} = –800µA, I_{OH} = –300µA I_{OH} = –80µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$			V V V
I_{IL}	Logical 0 input current, ports 1, 2, 3[7]	V_{IN} = 0.45V			–50	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3[7]	See note 4			–650	µA
I_{LI}	Input leakage current, port 0	$V_{IN} = V_{IL}$ or V_{IH}			±10	µA
I_{CC}	Power supply current (does not include AI_{CC}):[7] Active mode @ 16MHz[5] Idle mode @ 16MHz Power down mode	See note 6		 11.5 1.3 3	 25 5 50	 mA mA µA
R_{RST}	Internal reset pull-down resistor		50		300	kΩ
C_{IO}	Pin capacitance (I/O pins only)				10	pF

NOTES:
1. Typical ratings are not guaranteed. The values listed are at room temperature, 5V.
2. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
3. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
4. Pins of ports 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
5. I_{CC}MAX at other frequencies is given by: Active mode; I_{CC}MAX = 1.43 × FREQ + 1.90: Idle mode; I_{CC}MAX = 0.14 × FREQ +2.31, where FREQ is the external oscillator frequency in MHz. I_{CC}MAX is given in mA. See Figure 12.
6. See Figures 13 through 16 for I_{CC} test conditions.
7. These values apply only to T_{amb} = 0°C to +70°C. For T_{amb} = –40°C to +85°C. See table on previous page.

AC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or −40°C to +85°C, V_{CC} = 5V ±10% (87C550), V_{CC} = 5V ±20% (80/83C550), V_{SS} = 0V[1, 2]

SYMBOL	FIGURE	PARAMETER	16MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	5	Oscillator frequency: **Speed Versions** S8XC550 Exx			3.5	16	MHz
t_{LHLL}	5	ALE pulse width	85		$2t_{CLCL}-40$		ns
t_{AVLL}	5	Address valid to ALE low	7		$t_{CLCL}-55$		ns
t_{LLAX}	5	Address hold after ALE low	27		$t_{CLCL}-35$		ns
t_{LLIV}	5	ALE low to valid instruction in		150		$4t_{CLCL}-100$	ns
t_{LLPL}	5	ALE low to $\overline{PSEN}$ low	22		$t_{CLCL}-40$		ns
t_{PLPH}	5	$\overline{PSEN}$ pulse width	142		$3t_{CLCL}-45$		ns
t_{PLIV}	5	$\overline{PSEN}$ low to valid instruction in		82		$3t_{CLCL}-105$	ns
t_{PXIX}	5	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	5	Input instruction float after $\overline{PSEN}$		37		$t_{CLCL}-25$	ns
t_{AVIV}	5	Address to valid instruction in		207		$5t_{CLCL}-105$	ns
t_{PLAZ}	5	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	6, 7	$\overline{RD}$ pulse width	275		$6t_{CLCL}-100$		ns
t_{WLWH}	6, 7	$\overline{WR}$ pulse width	275		$6t_{CLCL}-100$		ns
t_{RLDV}	6, 7	$\overline{RD}$ low to valid data in		212		$5t_{CLCL}-165$	ns
t_{RHDX}	6, 7	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	6, 7	Data float after $\overline{RD}$		55		$2t_{CLCL}-70$	ns
t_{LLDV}	6, 7	ALE low to valid data in		350		$8t_{CLCL}-150$	ns
t_{AVDV}	6, 7	Address to valid data in		397		$9t_{CLCL}-165$	ns
t_{LLWL}	6, 7	ALE low to $\overline{RD}$ or $\overline{WR}$ low	137	247	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	6, 7	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	120		$4t_{CLCL}-130$		ns
t_{QVWX}	6, 7	Data valid to $\overline{WR}$ transition	12		$t_{CLCL}-50$		ns
t_{WHQX}	6, 7	Data hold after $\overline{WR}$	12		$t_{CLCL}-50$		ns
t_{RLAZ}	6, 7	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	6, 7	$\overline{RD}$ or $\overline{WR}$ high to ALE high	22	102	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	9	High time	20		20		ns
t_{CLCX}	9	Low time	20		20		ns
t_{CLCH}	9	Rise time		20		20	ns
t_{CHCL}	9	Fall time		20		20	ns
Shift Register							
t_{XLXL}	8	Serial port clock cycle time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	8	Output data setup to clock rising edge	492		$10t_{CLCL}-133$		ns
t_{XHQX}	8	Output data hold after clock rising edge	8		$2t_{CLCL}-117$		ns
t_{XHDX}	8	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	8	Clock rising edge to input data valid		492		$10t_{CLCL}-133$	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

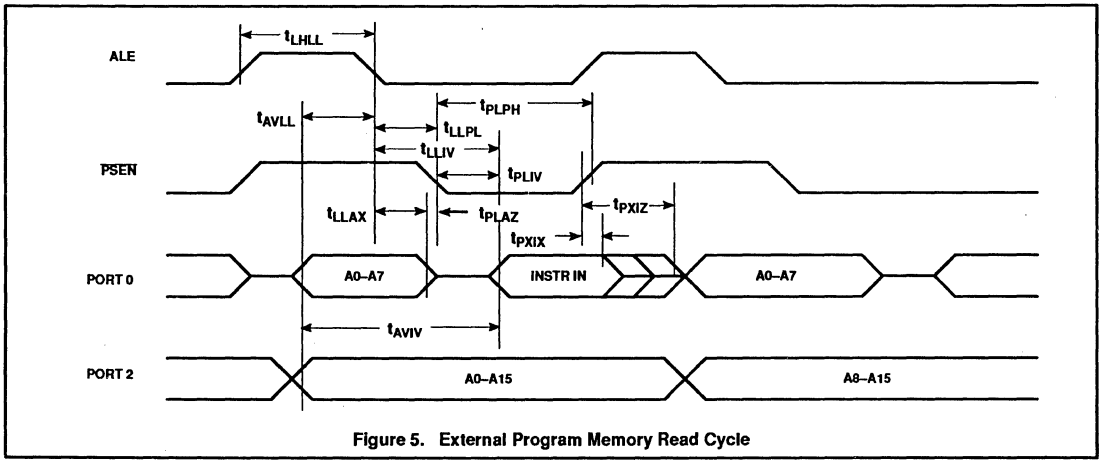


Figure 5. External Program Memory Read Cycle

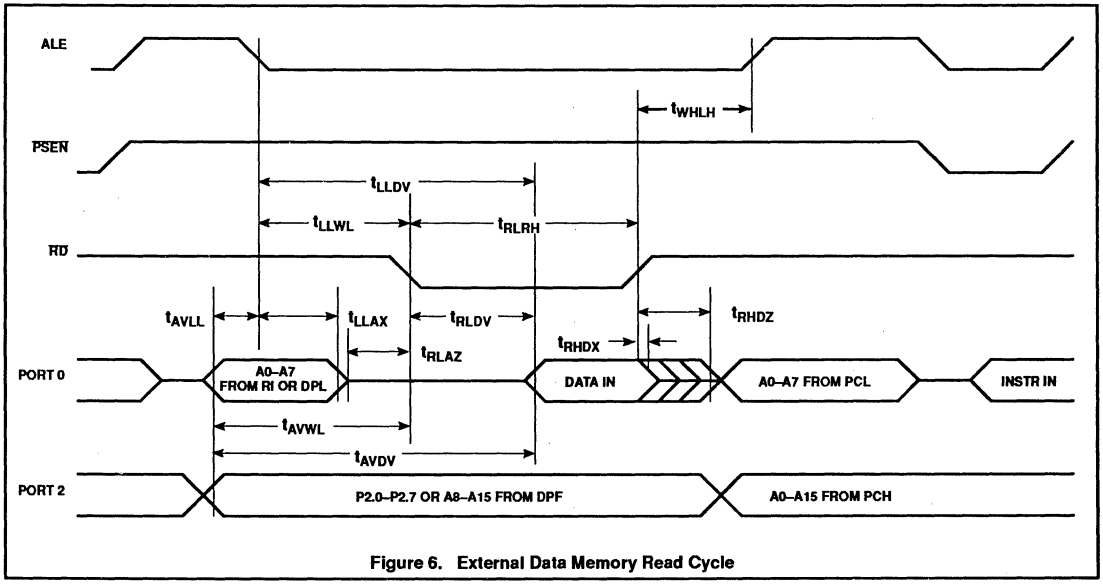


Figure 6. External Data Memory Read Cycle

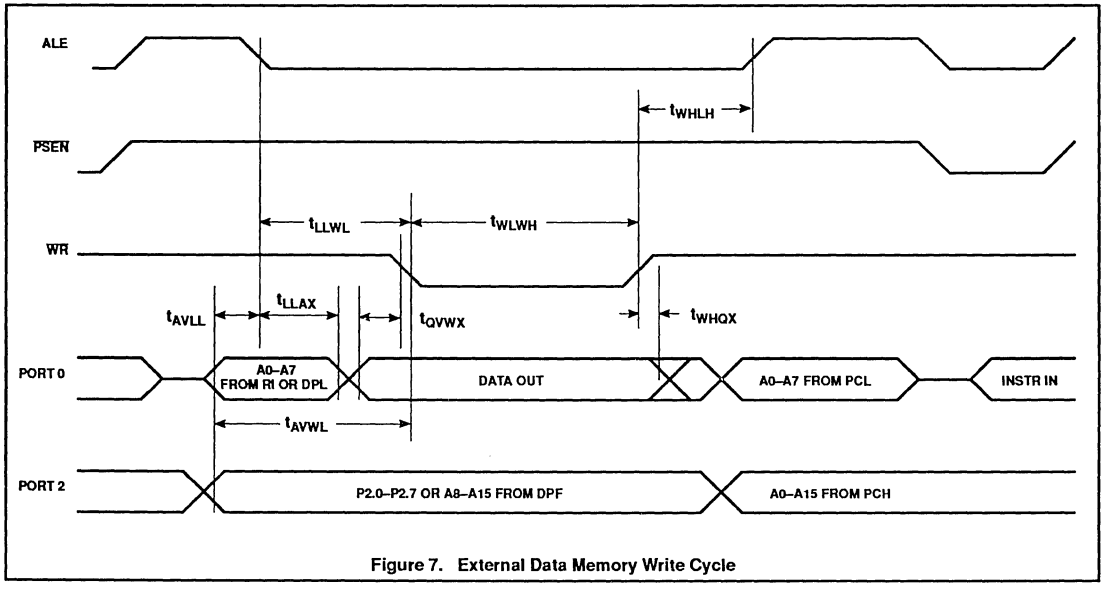


Figure 7. External Data Memory Write Cycle

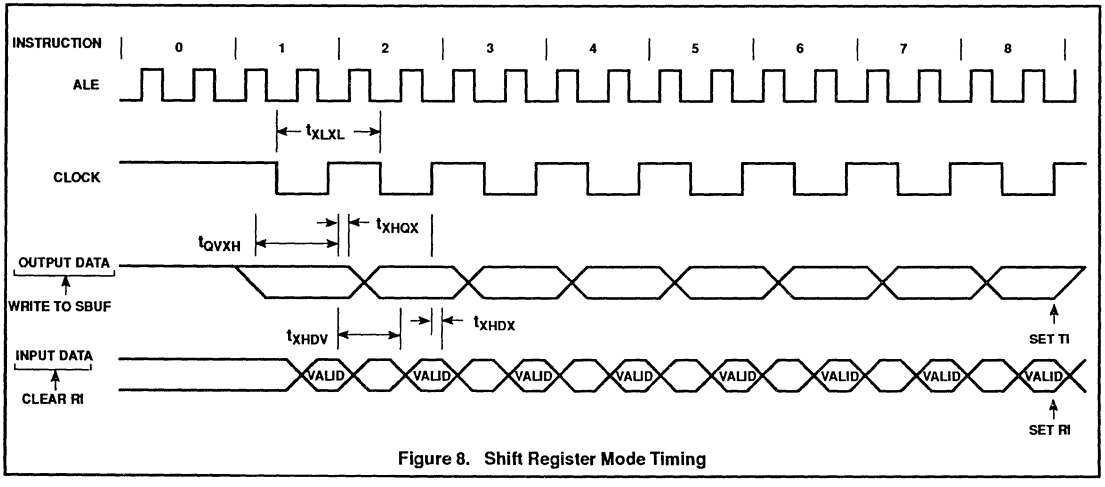


Figure 8. Shift Register Mode Timing

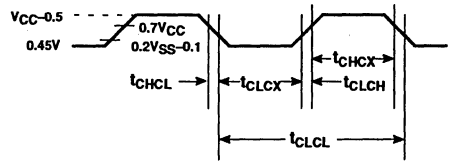
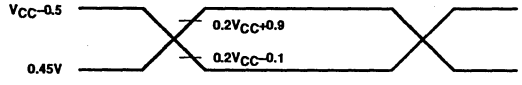
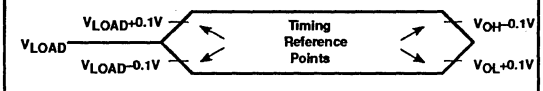


Figure 9. External Clock Drive



NOTE:
AC inputs during testing are driven at V_{CC} –0.5 for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at V_{IH} min for a logic '1' and V_{IL} for a logic '0'.

Figure 10. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 11. Float Waveform

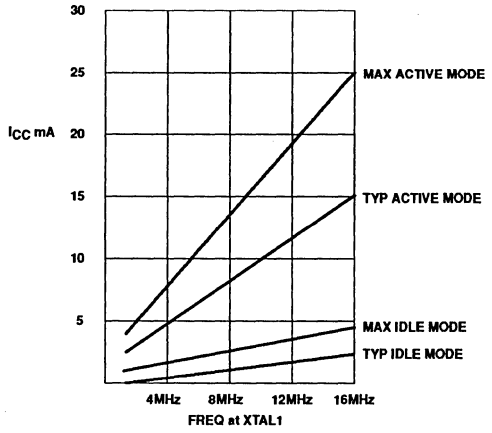


Figure 12. I_{CC} vs. FREQ
Valid only within frequency specifications of the device under test

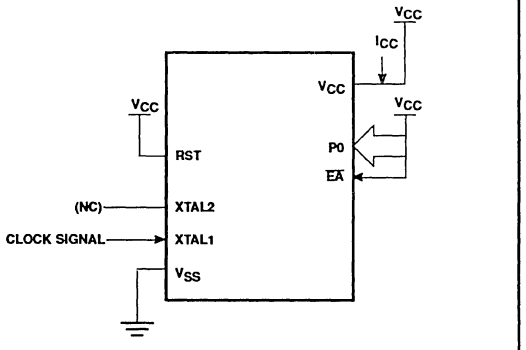


Figure 13. I_{CC} Test Condition, Active Mode
All other pins are disconnected

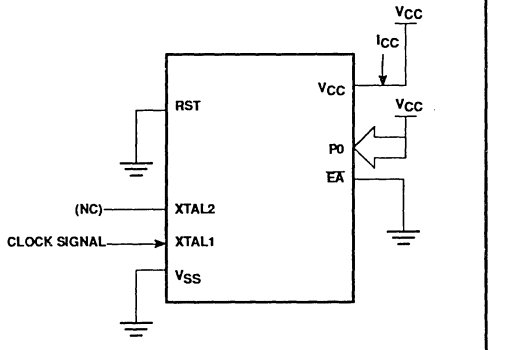


Figure 14. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

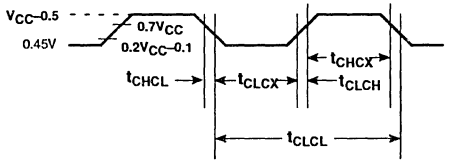


Figure 15. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

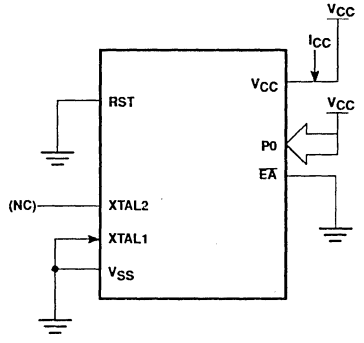


Figure 16. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected.
V_{CC} = 2V to 5.5V.

EPROM CHARACTERISTICS

The 87C550 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C550 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an S87C550 manufactured by Philips.

Table 4 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 17 and 18. Figure 19 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 17. Note that the 87C550 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 2 and 3, as shown in Figure 17. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 1 and 2 specified in Table 4 are held at the 'Program Code Data' levels indicated in Table 4. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 18.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 1FH, using the 'Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the security bits, repeat the 25 pulse programming sequence using the 'Pgm Security Bit' levels. After one security bit is programmed, further programming of the code memory and encryption table is disabled. However, the other security bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If security bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 2 and 3 as shown in Figure 19. The other pins are held at the 'Verify Code Data' levels indicated in Table 4. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P1.0 and P1.1 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(031H) = 96H indicates S87C550

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 4, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 4. EPROM Programming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P1.1	P1.0
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm security bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm security bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:

1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.

* ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

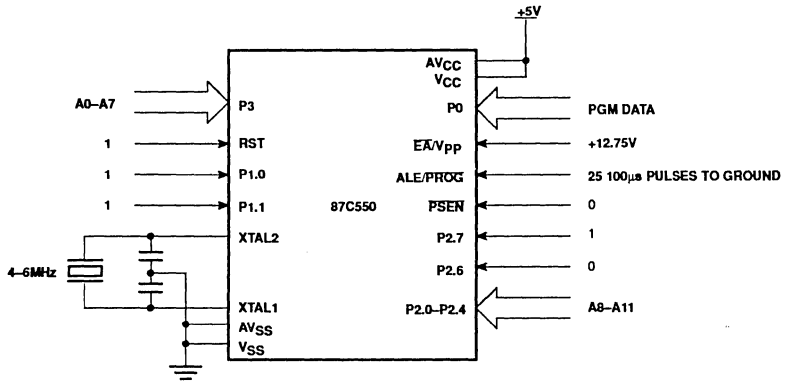


Figure 17. Programming Configuration

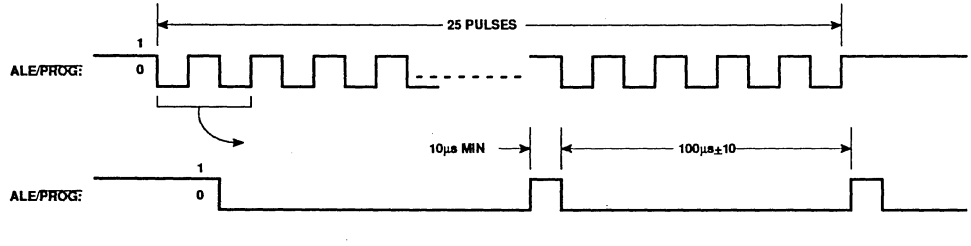


Figure 18. $\overline{\text{PROG}}$ Waveform

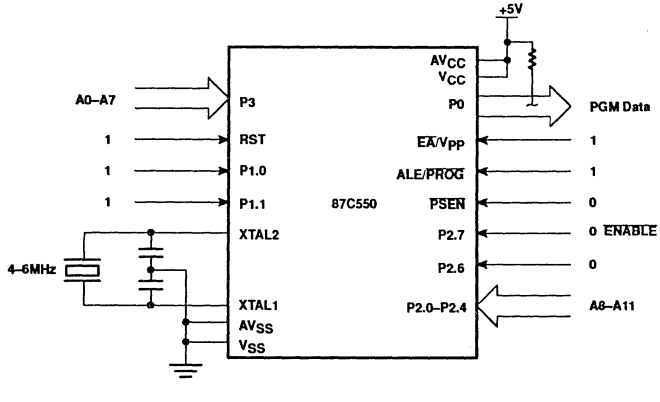


Figure 19. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_{amb} = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 20)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

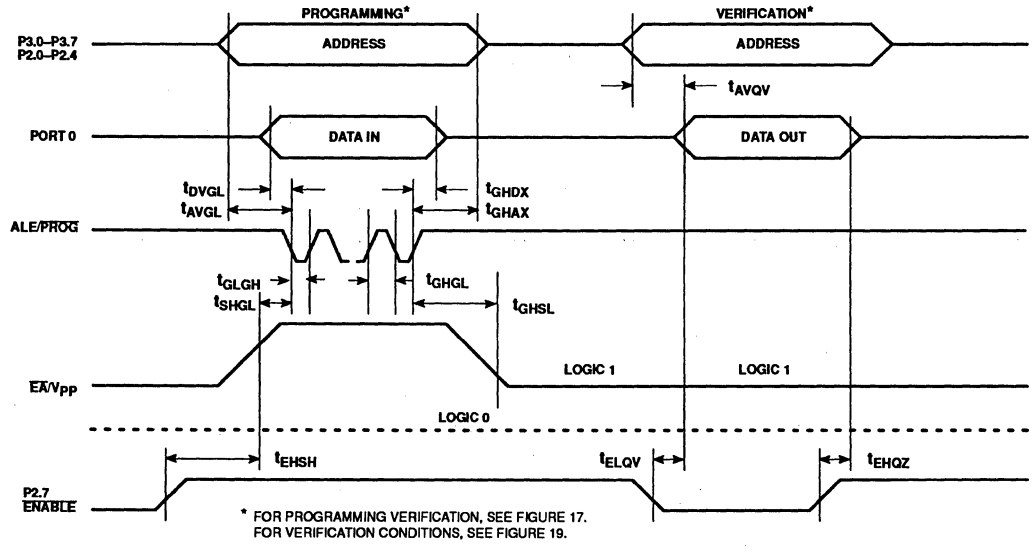


Figure 20. EPROM Programming and Verification

8XC552 OVERVIEW

The 8XC552 is a stand-alone high-performance microcontroller designed for use in real-time applications such as instrumentation, industrial control, and automotive control applications such as engine management and transmission control. The device provides, in addition to the 80C51 standard functions, a number of dedicated hardware functions for these applications.

The 8XC552 single-chip 8-bit microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 8XC552 uses the powerful instruction set of the 80C51. Additional special function registers are incorporated to control the on-chip peripherals. Three versions of the derivative exist although the generic term "8XC552" is used to refer to family members:

83C552: 8k bytes mask-programmable ROM, 256 bytes RAM

87C552: 8k bytes EPROM, 256 bytes RAM

80C552: ROMless version of the 83C552

The 8XC552 contains a nonvolatile 8k × 8 read-only program memory, a volatile 256 × 8 read/write data memory, five 8-bit I/O ports and one 8-bit input port, two 16-bit timer/event counters (identical to the timers of the 80C51), an additional 16-bit timer coupled to capture and compare latches, a fifteen-source, two-priority-level, nested interrupt structure, an 8-input ADC, a dual DAC pulse width modulated interface, two serial interfaces (UART and I^2C bus), a "watchdog" timer, and on-chip oscillator and timing circuits. For systems that require extra capability, the 8XC552 can be expanded using standard TTL compatible memories and logic

The 8XC552 has two software selectable modes of reduced activity for further power reduction—Idle and Power-down. The idle mode freezes the CPU and resets Timer T2 and the ADC and PWM circuitry but allows the other timers, RAM, serial ports, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to become inoperative.

83C562 OVERVIEW

The 83C562 has been derived from the 8XC552 with the following changes:

- The SIO1 (I^2C) interface has been omitted.
- The output of port lines P1.6 and P1.7 have a standard configuration instead of open drain.
- The resolution of the A/D converter is decreased from 10 bits to 8 bits.
- The time of an A/D conversion has decreased from 50 machine cycles to 24 machine cycles.

All other functions, pinning and packaging are unchanged.

This chapter of the users' guide can be used for the 83C562 by omitting or changing the following:

- Disregard the description of SIO1 (I^2C).
- The SFRs for the interface: S1ADR, S1DAT, S1STA, and S1CON are not implemented. The two SIO1 related flags ES1 in SFR IEN0 and PS1 in SFR IP0 are also not implemented. These two flag locations are undefined after RESET. The interrupt vector for SIO1 is not used.
- Port lines P1.6 and P1.7 are not open drain but have the same standard configuration and electrical characteristics as P1.0-P1.5. Port lines P1.6 and P1.7 have alternative functions.
- The A/D converter has a resolution of 8 bits instead of 10 bits and consequently the two high-order bits 6 and 7 of SFR ADCON are not implemented. These two locations are undefined after RESET. The 8-bit result of an A/D conversion is present in SFR ADCH. The result can always be calculated from the formula:

$$256 \times \frac{V_{IN} - AV_{ref-}}{AV_{ref+} - AV_{ref-}}$$

  The A/D conversion time is 24 machine cycles instead of 50 machine cycles, and the sampling time is 6 machine cycles instead of 8 machine cycles. The conversion time takes 3 machine cycles per bit.
- The serial I/O function SIO0 and its SFRs S0BUF and S0CON are renamed to SIO, SBUF, and SCON. The interrupt related flags ES0 and PS0 are renamed ES and PS. Interrupt source S0 is renamed S. The serial I/O function remains the same.

Differences From the 80C51

Program Memory

The 8XC552 contains 8k bytes of on-chip program memory which can be extended to 64k bytes with external memories (see Figure 1). When the $\overline{EA}$ pin is held high, the 8XC552 fetches instructions from internal ROM unless the address exceeds 1FFFH. Locations 2000H to FFFFH are fetched from external program memory. When the $\overline{EA}$ pin is held low, all instruction fetches are from external memory. ROM locations 0003H to 0073H are used by interrupt service routines.

Data Memory

The internal data memory is divided into 3 sections: the lower 128 bytes of RAM, the upper 128 bytes of RAM, and the 128-byte special function register areas. The lower 128 bytes of RAM are directly and indirectly addressable. While RAM locations 128 to 255 and the special function register area share the same address space, they are accessed through different addressing modes. RAM locations 128 to 255 are only indirectly addressable, and the special function registers are only directly addressable. All other aspects of the internal RAM are identical to the 8051.

The stack may be located anywhere in the internal RAM by loading the 8-bit stack pointer. Stack depth is 256 bytes maximum.

Special Function Registers

The special function registers (directly addressable only) contain all of the 8XC552 registers except the program counter and the four register banks. Most of the 56 special function registers are used to control the on-chip peripheral hardware. Other registers include arithmetic registers (ACC, B, PSW), stack pointer (SP), and data pointer registers (DHP, DPL). Sixteen of the SFRs contain 128 directly addressable bit locations. Table 14 lists the 8XC552's special function registers.

The standard 80C51 SFRs are present and function identically in the 8XC552 except where noted in the following sections.

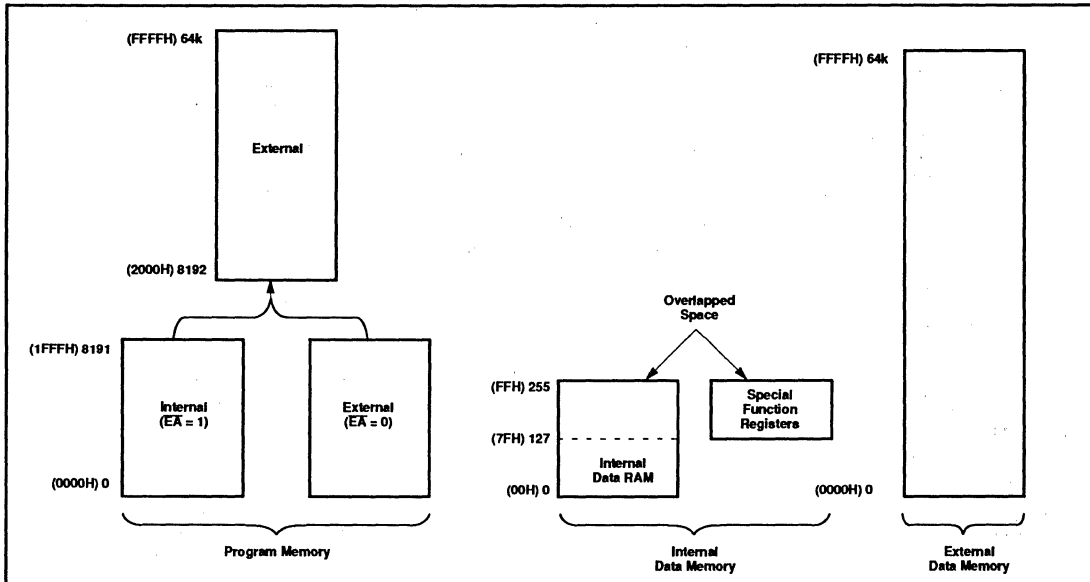


Figure 1. Memory Map

Timer T2

Timer T2 is a 16-bit timer consisting of two registers TMH2 (HIGH byte) and TML2 (LOW byte). The 16-bit timer/counter can be switched off or clocked via a prescaler from one of two sources: fosc/12 or an external signal. When Timer T2 is configured as a counter, the prescaler is clocked by an external signal on T2 (P1.4). A rising edge on T2 increments the prescaler, and the maximum repetition rate is one count per machine cycle (1MHz with a 12MHz oscillator).

The maximum repetition rate for Timer T2 is twice the maximum repetition rate for Timer 0 and Timer 1. T2 (P1.4) is sampled at S2P1 and again at S5P1 (i.e., twice per machine cycle). A rising edge is detected when T2 is LOW during one sample and HIGH during the next sample. To ensure that a rising edge is detected, the input signal must be LOW for at least 1/2 cycle and then HIGH for at least 1/2 cycle. If a rising edge is detected before the end of S2P1, the timer will be incremented during the following cycle; otherwise it will be incremented one cycle later. The prescaler has a programmable division factor of 1, 2, 4, or 8 and is cleared if its division factor or input source is changed, or if the timer/counter is reset.

Timer T2 may be read "on the fly" but possesses no extra read latches, and software precautions may have to be taken to avoid misinterpretation in the event of an overflow from least to most significant byte while Timer T2 is being read. Timer T2 is not loadable and is reset by the RST signal or by a rising edge on the input signal RT2, if enabled. RT2 is enabled by setting bit T2ER (TM2CON.5).

When the least significant byte of the timer overflows or when a 16-bit overflow occurs, an interrupt request may be generated. Either or both of these overflows can be programmed to request an interrupt. In both cases, the interrupt vector will be the same. When the lower byte (TML2) overflows, flag T2B0 (TM2CON) is set and flag T20V (TM2IR) is set when TMH2 overflows. These flags are set one cycle after an overflow occurs. Note that when T20V is set, T2B0 will also be set. To enable the byte overflow interrupt, bits ET2 (IEN1.7, enable overflow interrupt, see Figure 2) and T2IS0 (TM2CON.6, byte overflow interrupt select) must be set. Bit TWB0 (TM2CON.4) is the Timer T2 byte overflow flag.

To enable the 16-bit overflow interrupt, bits ET2 (IE1.7, enable overflow interrupt) and T2IS1 (TM2CON.7, 16-bit overflow interrupt select) must be set. Bit T2OV (TM2IR.7) is the Timer T2 16-bit overflow flag. All interrupt flags must be reset by software. To enable both byte and 16-bit overflow, T2IS0 and T2IS1 must be set and two interrupt service routines are required. A test on the overflow flags indicates which routine must be executed. For each routine, only the corresponding overflow flag must be cleared.

Timer T2 may be reset by a rising edge on RT2 (P1.5) if the Timer T2 external reset enable bit (T2ER) in T2CON is set. This reset also clears the prescaler. In the idle mode, the timer/counter and prescaler are reset and halted. Timer T2 is controlled by the TM2CON special function register (see Figure 3).

Table 1. 8XC552 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
ADCH#	A/D converter high	C6H									xxxxxxxxB
ADCON#	Adc control	C5H	ADC.1	ADC.0	ADEX	ADCI	ADCS	AADR2	AADR1	AADR0	xx000000B
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
CTCON#	Capture control	EBH	CTN3	CTP3	CTN2	CTP2	CTN1	CTP1	CTN0	CTP0	00H
CTH3#	Capture high 3	CFH									xxxxxxxxB
CTH2#	Capture high 2	CEH									xxxxxxxxB
CTH1#	Capture high 1	CDH									xxxxxxxxB
CTH0#	Capture high 0	CCH									xxxxxxxxB
CMH2#	Compare high 2	CBH									00H
CMH1#	Compare high 1	CAH									00H
CMH0#	Compare high 0	C9H									00H
CTL3#	Capture low 3	AFH									xxxxxxxxB
CTL2#	Capture low 2	AEH									xxxxxxxxB
CTL1#	Capture low 1	ADH									xxxxxxxxB
CTL0#	Capture low 0	ACH									xxxxxxxxB
CML2#	Compare low 2	ABH									00H
CML1#	Compare low 1	AAH									00H
CML0#	Compare low 0	A9H									00H
DPTR:	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IEN0*#	Interrupt enable 0	A8H	EA	EAD	ES1	ES0	ET1	EX1	ET0	EX0	00H
			EF	EE	ED	EC	EB	EA	E9	E8	
IEN1*#	Interrupt enable 1	E8H	ET2	ECM2	ECM1	ECM0	ECT3	ECT2	ECT1	ECT0	00H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP0*#	Interrupt priority 0	B8H	–	PAD	PS1	PS0	PT1	PX1	PT0	PX0	x0000000B
			FF	FE	FD	FC	FB	FA	F9	F8	
IP1*#	Interrupt priority 1	F8H	PT2	PCM2	PCM1	PCM0	PCT3	PCT2	PCT1	PCT0	00H
P5#	Port 5	C4H	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	xxxxxxxxB
			C7	C6	C5	C4	C3	C2	C1	C0	
P4#	Port 4	C0H	CMT1	CMT0	CMSR5	CMSR4	CMSR3	CMSR2	CMSR1	CMSR0	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	$\overline{RD}$	$\overline{WR}$	T1	T0	$\overline{INT1}$	$\overline{INT0}$	TXD	RXD	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	A15	A14	A13	A12	A11	A10	A9	A8	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	SDA	SCL	RT2	T2	CT3I	CT2I	CT1I	CT0I	FFH
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
PCON#	Power control	87H	SMOD	–	–	WLE	GF1	GF0	PD	IDL	00xx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	F1	P	00H

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.

Table 1. 8XC552 Special Function Registers (Continued)

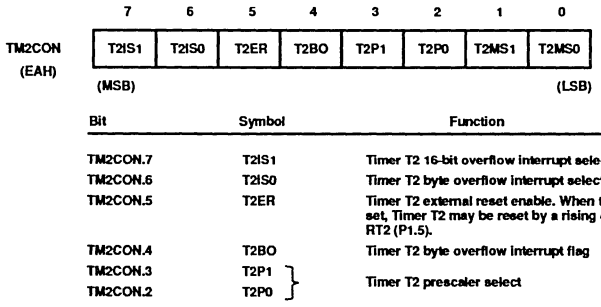
SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
PWMP#	PWM prescaler	FEH									00H
PWM1#	PWM register 1	FDH									00H
PWM0#	PWM register 0	FCH									00H
RTE#	Reset/toggle enable	EFH	TP47	TP46	RP45	RP44	RP43	RP42	RP41	RP40	00H
SP	Stack pointer	81H									07H
S0BUF	Serial 0 data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
S0CON*	Serial 0 control	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
S1ADR#	Serial 1 address	DBH	SLAVE ADDRESS							GC	00H
SIDAT#	Serial 1 data	DAH									00H
S1STA#	Serial 1 status	D9H	SC4	SC3	SC2	SC1	SC0	0	0	0	F8H
			DF	DE	DD	DC	DB	DA	D9	D8	
SICON#*	Serial 1 control	D8H	CR2	ENS1	STA	ST0	SI	AA	CR1	CR0	00H
STE#	Set enable	EEH	TG47	TG46	SP45	SP44	SP43	SP42	SP41	SP40	C0H
TH1	Timer high 1	8DH									00H
TH0	Timer high 0	8CH									00H
TL1	Timer low 1	8BH									00H
TL0	Timer low 0	8AH									00H
TMH2#	Timer high 2	EDH									00H
TML2#	Timer low 2	ECH									00H
TMOD	Timer mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
TM2CON#	Timer 2 control	EAH	T2IS1	T2IS0	T2ER	T2B0	T2P1	T2P0	T2MS1	T2MS0	00H
			CF	CE	CD	CC	CB	CA	C9	C8	
TM2IR#	Timer 2 int flag reg	C8H	T20V	CMI2	CMI1	CMI0	CTI3	CTI2	CTI1	CTI0	00H
T3#	Timer 3	FFH									00H

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.

	7	6	5	4	3	2	1	0
IEN1 (E8H	ET2	ECM2	ECM1	ECM0	ECT3	ECT2	ECT1	ECT0
	(MSB)							(LSB)

Bit	Symbol	Function
IEN1.7	ET2	Enable Timer T2 overflow interrupt(s)
IEN1.6	ECM2	Enable T2 Comparator 2 interrupt
IEN1.5	ECM1	Enable T2 Comparator 1 interrupt
IEN1.4	ECM0	Enable T2 Comparator 0 interrupt
IEN1.3	ECT3	Enable T2 Capture register 3 interrupt
IEN1.2	ECT2	Enable T2 Capture register 2 interrupt
IEN1.1	ECT1	Enable T2 Capture register 1 interrupt
IEN1.0	ECT0	Enable T2 Capture register 0 interrupt

Figure 2. Timer T2 Interrupt Enable Register (IEN1)



	7	6	5	4	3	2	1	0
TM2CON (EAH)	T2IS1	T2IS0	T2ER	T2BO	T2P1	T2P0	T2MS1	T2MS0
	(MSB)							(LSB)

Bit	Symbol	Function
TM2CON.7	T2IS1	Timer T2 16-bit overflow interrupt select
TM2CON.6	T2IS0	Timer T2 byte overflow interrupt select
TM2CON.5	T2ER	Timer T2 external reset enable. When this bit is set, Timer T2 may be reset by a rising edge on RT2 (P1.5).
TM2CON.4	T2BO	Timer T2 byte overflow interrupt flag
TM2CON.3	T2P1	Timer T2 prescaler select
TM2CON.2	T2P0	

T2P1	T2P0	Timer T2 Clock
0	0	Clock source
0	1	Clock source/2
1	0	Clock source/4
1	1	Clock source/8

Bit	Symbol	Function
TM2CON.1	T2MS1	Timer T2 mode select
TM2CON.2	T2MS0	

T2MS1	T2MS0	Mode Selected
0	0	Timer T2 halted (off)
0	1	T2 clock source = $f_{osc}/12$
1	0	Test mode; do not use
1	1	T2 clock source = pin T2

Figure 3. T2 Control Register (TM2CON)

Timer T2 Extension: When a 12MHz oscillator is used, a 16-bit overflow on Timer T2 occurs every 65.5, 131, 262, or 524 ms, depending on the prescaler division ratio; i.e., the maximum cycle time is approximately 0.5 seconds. In applications where cycle times are greater than 0.5 seconds, it is necessary to extend Timer T2. This is achieved by selecting fosc/12 as the clock source (set T2MS0, reset T2MS1), setting the prescaler division ration to 1/8 (set T2P0, set T2P1), disabling the byte overflow interrupt (reset T2IS0) and enabling the 16-bit overflow interrupt (set T2IS1). The following software routine is written for a three-byte extension which gives a maximum cycle time of approximately 2400 hours.

```
OVINT: PUSH   ACC     ;save accumulator
       PUSH   PSW     ;save status
       INC    TIMEX1  ;increment first
                      ;byte (low order)
                      ;of extended timer
       MOV    A,TIMEX1
       JNZ    INTEX   ;jump to INTEX if
                      ;there is no
                      ;overflow
       INC    TIMEX2  ;increment second
                      ;byte
       MOV    A,TIMEX2
       JNZ    INTEX   ;jump to INTEX if
                      ;there is no
                      ;overflow
       INC    TIMEX3  ;increment third
                      ;byte (high order)

INTEX: CLR    T2OV    ;reset interrupt
                      ;flag
       POP    PSW     ;restore status
       POP    ACC     ;restore
                      accumulator
       RETI           ;return from
                      ;interrupt
```

Timer T2, Capture and Compare Logic: Timer T2 is connected to four 16-bit capture registers and three 16-bit compare registers. A capture register may be used to capture the contents of Timer T2 when a transition occurs on its corresponding input pin. A compare register may be used to set, reset, or toggle port 4 output pins at certain pre-programmable time intervals.

The combination of Timer T2 and the capture and compare logic is very powerful in applications involving rotating machinery, automotive injection systems, etc. Timer T2 and the capture and compare logic are shown in Figure 4.

Capture Logic: The four 16-bit capture registers that Timer T2 is connected to are: CT0, CT1, CT2, and CT3. These registers are loaded with the contents of Timer T2, and an interrupt is requested upon receipt of the input signals CT0I, CT1I, CT2I, or CT3I. These input signals are shared with port 1. The four interrupt flags are in the Timer T2 interrupt register (TM2IR special function register). If the capture facility is not required, these inputs can be regarded as additional external interrupt inputs.

Using the capture control register CTCON (see Figure 5), these inputs may capture on a rising edge, a falling edge, or on either a rising or falling edge. The inputs are sampled during S1P1 of each cycle. When a selected edge is detected, the contents of Timer T2 are captured at the end of the cycle.

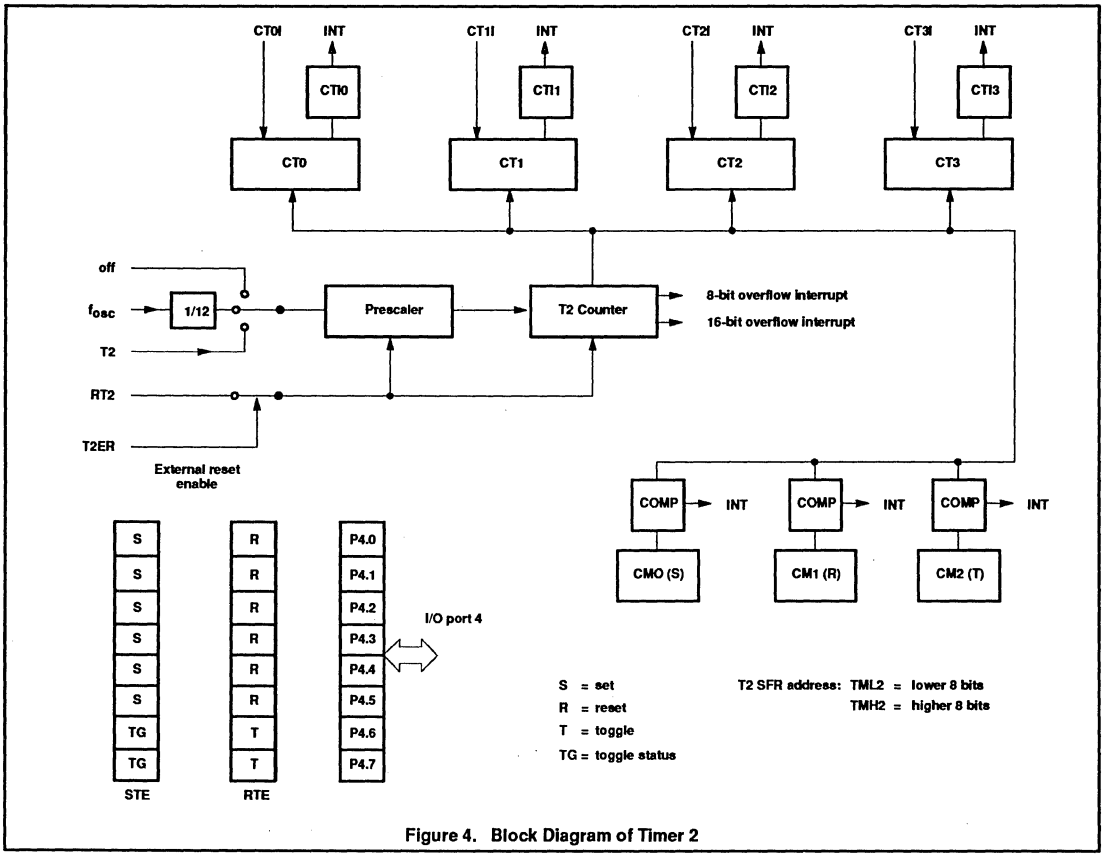


Figure 4. Block Diagram of Timer 2

Measuring Time Intervals Using Capture Registers: When a recurring external event is represented in the form of rising or falling edges on one of the four capture pins, the time between two events can be measured using Timer T2 and a capture register. When an event occurs, the contents of Timer T2 are copied into the relevant capture register and an interrupt request is generated. The interrupt service routine may then compute the interval time if it knows the previous contents of Timer T2 when the last event occurred. With a 12MHz oscillator, Timer T2 can be programmed to overflow every 524ms. When event interval times are shorter than this, computing the interval time is simple, and the interrupt service routine is short. For longer interval times, the Timer T2 extension routine may be used.

Compare Logic: Each time Timer T2 is incremented, the contents of the three 16-bit compare registers CM0, CM1, and CM2 are compared with the new counter value of Timer T2. When a match is found, the corresponding interrupt flag in TM2IR is set at the end of the following cycle. When a match with CM0 occurs, the controller sets bits 0-5 of port 4 if the corresponding bits of the set enable register STE are at logic 1.

When a match with CM1 occurs, the controller resets bits 0-5 of port 4 if the corresponding bits of the reset/toggle enable register RTE are at logic 1 (see Figure 6 for RTE register function). If RTE is "0", then P4.n is not affected by a match between CM1 or CM2 and Timer 2. When a match with CM2 occurs, the controller "toggles" bits 6 and 7 of port 4 if the corresponding bits of the RTE are at logic 1. The port latches of bits 6 and 7 are not toggled. Two additional flip-flops store the last operation, and it is these flip-flops that are toggled.

Thus, if the current operation is "set," the next operation will be "reset" even if the port latch is reset by software before the "reset" operation occurs. The first "toggle" after a chip RESET will set the port latch. The contents of these two flip-flops can be read at STE.6 and STE.7 (corresponding to P4.6 and P4.7, respectively). Bits STE.6 and STE.7 are read only (see Figure 7 for STE register function). A logic 1 indicates that the next toggle will set the port latch; a logic 0 indicates that the next toggle will reset the port latch. CM0, CM1, and CM2 are reset by the RST signal.

The modified port latch information appears at the port pin during S5P1 of the cycle following the cycle in which a match occurred. If the port is modified by software, the outputs change during S1P1 of the following cycle. Each port 4 bit can be set or reset by software at any time. A hardware modification resulting from a comparator match takes precedence over a software modification in the same cycle. When the comparator results require a "set" and a "reset" at the same time, the port latch will be reset.

Timer T2 Interrupt Flag Register TM2IR: Eight of the nine Timer T2 interrupt flags are located in special function register TM2IR (see Figure 8). The ninth flag is TM2CON.4.

The CT0I and CT1I flags are set during S4 of the cycle in which the contents of Timer T2 are captured. CT0I is scanned by the interrupt logic during S2, and CT1I is scanned during S3. CT2I and CT3I are set during S6 and are scanned during S4 and S5. The associated interrupt requests are recognized during the following cycle. If these flags are polled, a transition at CT0I or CT1I will be recognized one cycle before a transition on CT2I or CT3I since registers are read during S5. The CMI0, CMI1, and CMI2 flags are set during S6 of the cycle following a match. CMI0 is scanned by the interrupt logic during S2; CMI1 and CMI2 are scanned during S3 and S4. A match will be recognized by the interrupt logic (or by polling the flags) two cycles after the match takes place.

The 16-bit overflow flag (T2OV) and the byte overflow flag (T2BO) are set during S6 of the cycle in which the overflow occurs. These flags are recognized by the interrupt logic during the next cycle.

Special function register IP1 (Figure 8) is used to determine the Timer T2 interrupt priority. Setting a bit high gives that function a high priority, and setting a bit low gives the function a low priority. The functions controlled by the various bits of the IP1 register are shown in Figure 8.

	7	6	5	4	3	2	1	0
CTCON (E8H)	CTN3	CTP3	CTN2	CTP2	CTN1	CTP1	CTN0	CTP0
	(MSB)							(LSB)

Bit	Symbol	Capture/Interrupt on:
CTCON.7	CTN3	Capture Register 3 triggered by a falling edge on CT3I
CTCON.6	CTP3	Capture Register 3 triggered by a rising edge on CT3I
CTCON.5	CTN2	Capture Register 2 triggered by a falling edge on CT2I
CTCON.4	CTP2	Capture Register 2 triggered by a rising edge on CT2I
CTCON.3	CTN1	Capture Register 1 triggered by a falling edge on CT1I
CTCON.2	CTP1	Capture Register 1 triggered by a rising edge on CT1I
CTCON.1	CTN0	Capture Register 0 triggered by a falling edge on CT0I
CTCON.0	CTP0	Capture Register 0 triggered by a rising edge on CT0I

Figure 5. Capture Control Register (CTCON)

	7	6	5	4	3	2	1	0
RTE (EFH)	TP47	TP46	RP45	RP44	RP43	RP42	RO41	RP40
	(MSB)							(LSB)

Bit	Symbol	Function
RTE.7	TP47	If "1" then P4.7 toggles on a match between CM2 and Timer T2
RTE.6	TP46	If "1" then P4.6 toggles on a match between CM2 and Timer T2
RTE.5	RP45	If "1" then P4.5 is reset on a match between CM1 and Timer T2
RTE.4	RP44	If "1" then P4.4 is reset on a match between CM1 and Timer T2
RTE.3	RP43	If "1" then P4.3 is reset on a match between CM1 and Timer T2
RTE.2	RP42	If "1" then P4.2 is reset on a match between CM1 and Timer T2
RTE.1	RP41	If "1" then P4.1 is reset on a match between CM1 and Timer T2
RTE.0	RP40	If "1" then P4.0 is reset on a match between CM1 and Timer T2

Figure 6. Reset/Toggle Enable Register (RTE)

	7	6	5	4	3	2	1	0
STE (EEH)	TG47	TG46	SP45	SP44	SP43	SP42	SP41	SP40
	(MSB)							(LSB)

Bit	Symbol	Function
STE.7	TG47	Toggle flip-flops
STE.6	TG46	Toggle flip-flops
STE.5	SP45	If "1" then P4.5 is set on a match between CM0 and Timer T2
STE.4	SP44	If "1" then P4.4 is set on a match between CM0 and Timer T2
STE.3	SP43	If "1" then P4.3 is set on a match between CM0 and Timer T2
STE.2	SP42	If "1" then P4.2 is set on a match between CM0 and Timer T2
STE.1	SP41	If "1" then P4.1 is set on a match between CM0 and Timer T2
STE.0	SP40	If "1" then P4.0 is set on a match between CM0 and Timer T2

Figure 7. Set Enable Register (STE)

	7	6	5	4	3	2	1	0
TM2IR (C8H)	T2OV	CMI2	CMI1	CMI0	CTI3	CTI2	CTI1	CTI0
	(MSB)							(LSB)

Bit	Symbol	Function
TM2IR.7	T2OV	Timer T2 16-bit overflow interrupt flag
TM2IR.6	CMI2	CM2 interrupt flag
TM2IR.5	CMI1	CM1 interrupt flag
TM2IR.4	CMI0	CM0 interrupt flag
TM2IR.3	CTI3	CT3 interrupt flag
TM2IR.2	CTI2	CT2 interrupt flag
TM2IR.1	CTI1	CT1 interrupt flag
TM2IR.0	CTI0	CT0 interrupt flag

Interrupt Flag Register (TM2IR)

	7	6	5	4	3	2	1	0
IP1 (F8H)	PT2	PCM2	PCM1	PCM0	PCT3	PCT2	PCT1	PCT0
	(MSB)							(LSB)

Bit	Symbol	Function
IP1.7	PT2	Timer T2 overflow interrupt(s) priority level
IP1.6	PCM2	Timer T2 comparator 2 interrupt priority level
IP1.5	PCM1	Timer T2 comparator 1 interrupt priority level
IP1.4	PCM0	Timer T2 comparator 0 interrupt priority level
IP1.3	PCT3	Timer T2 capture register 3 interrupt priority level
IP1.2	PCT2	Timer T2 capture register 2 interrupt priority level
IP1.1	PCT1	Timer T2 capture register 1 interrupt priority level
IP1.0	PCT0	Timer T2 capture register 0 interrupt priority level

Timer 2 Interrupt Priority Register (IP1)

Figure 8. Interrupt Flag Register (TM2IR) and Timer T2 Interrupt Priority Register (IP1)

Timer T3, The Watchdog Timer

In addition to Timer T2 and the standard timers, a watchdog timer is also incorporated on the 8XC552. The purpose of a watchdog timer is to reset the microcontroller if it enters erroneous processor states (possibly caused by electrical noise or RFI) within a reasonable period of time. An analogy is the "dead man's handle" in railway locomotives. When enabled, the watchdog circuitry will generate a system reset if the user program fails to reload the watchdog timer within a specified length of time known as the "watchdog interval."

Watchdog Circuit Description: The watchdog timer (Timer T3) consists of an 8-bit timer with an 11-bit prescaler as shown in Figure 9. The prescaler is fed with a signal whose frequency is 1/12 the oscillator frequency (1MHz with a 12MHz oscillator). The 8-bit timer is incremented every "t" seconds, where:

$t = 12 \times 2048 \times 1/fosc$
(= 1.5ms at fosc = 16MHz;
= 1ms at fosc = 24MHz)

If the 8-bit timer overflows, a short internal reset pulse is generated which will reset the 8XC552. A short output reset pulse is also generated at the RST pin. This short output pulse (3 machine cycles) may be destroyed if the RST pin is connected to a capacitor. This would not, however, affect the internal reset operation.

Watchdog operation is activated when external pin $\overline{EW}$ is tied low. When $\overline{EW}$ is tied low, it is impossible to disable the watchdog operation by software.

How to Operate the Watchdog Timer: The watchdog timer has to be reloaded within periods that are shorter than the programmed watchdog interval; otherwise the watchdog timer will overflow and a system reset will be generated. The user program must therefore continually execute sections of code which reload the watchdog timer. The period of time elapsed between execution of these sections of code must never exceed the watchdog interval. When using a 16MHz oscillator, the watchdog interval is programmable between 1.5ms and 392ms. When using a 24MHz oscillator, the watchdog interval is programmable between 1ms and 255ms.

In order to prepare software for watchdog operation, a programmer should first determine how long his system can sustain an erroneous processor state. The result will be the maximum watchdog interval. As the maximum watchdog interval becomes shorter, it becomes more difficult for the programmer to ensure that the user program always reloads the watchdog timer within the watchdog interval, and thus it becomes more difficult to implement watchdog operation.

The programmer must now partition the software in such a way that reloading of the watchdog is carried out in accordance with the above requirements. The programmer must determine the execution times of all software modules. The effect of possible conditional branches, subroutines, external and internal interrupts must all be taken into account. Since it may be very difficult to evaluate the execution times of some sections of code, the programmer should use worst case estimations. In any event, the programmer must make sure that the watchdog is not activated during normal operation.

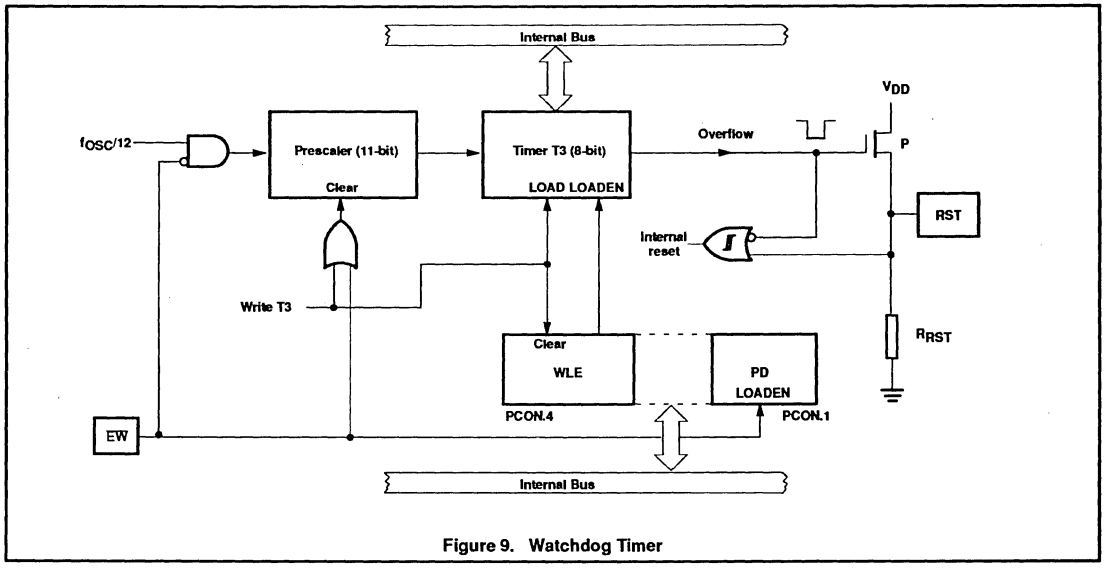


Figure 9. Watchdog Timer

The watchdog timer is reloaded in two stages in order to prevent erroneous software from reloading the watchdog. First PCON.4 (WLE) must be set. The T3 may be loaded. When T3 is loaded, PCON.4 (WLE) is automatically reset. T3 cannot be loaded if PCON.4 (WLE) is reset. Reload code may be put in a subroutine as it is called frequently. Since Timer T3 is an up-counter, a reload value of 00H gives the maximum watchdog interval (510ms with a 12MHz oscillator), and a reload value of 0FFH gives the minimum watchdog interval (2ms with a 12MHz oscillator).

In the idle mode, the watchdog circuitry remains active. When watchdog operation is implemented, the power-down mode cannot be used since both states are contradictory. Thus, when watchdog operation is enabled by tying external pin $\overline{EW}$ low, it is impossible to enter the power-down mode, and an attempt to set the power-down bit (PCON.1) will have no effect. PCON.1 will remain at logic 0.

During the early stages of software development/debugging, the watchdog may be disabled by tying the $\overline{EW}$ pin high. At a later stage, $\overline{EW}$ may be tied low to complete the debugging process.

Watchdog Software Example: The following example shows how watchdog operation might be handled in a user program.

```
;at the program start:

T3          EQU  0FFH ;address of
                      ;watchdog
                      ;timer T3
PCON        EQU  087H ;address of
                      ;PCON SFR
WATCH-INTV  EQU  156  ;watchdog
                      ;interval
                      ;(e.g., 2x100ms)

;to be inserted at each watchdog reload
;location within the user program:

LCALL WATCHDOG

;watchdog service routine:

WATCHDOG: ORL  PCON,#10H ;set
                         ;condition
                         ;flag
                         (PCON.4)
          MOV T3,WATCH-INV ;load T3
                           ;with
                           ;watchdog
                           ;interval
          RET
```

If it is possible for this subroutine to be called in an erroneous state, then the condition flag WLE should be set at different parts of the main program.

Serial I/O

The 8XC552 is equipped with two independent serial ports: SIO0 and SIO1. SIO0 is a full duplex UART port and is identical to the 80C51 serial port. SIO1 accommodates the I^2C bus.

SIO0: SIO0 is a full duplex serial I/O port identical to that on the 80C51. Its operation is the same, including the use of timer 1 as a baud rate generator.

SIO1, I2C Serial I/O: The I^2C bus uses two wires (SDA and SCL) to transfer information between devices connected to the bus. The main features of the bus are:

- Bidirectional data transfer between masters and slaves
- Multimaster bus (no central master)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer
- The I^2C bus may be used for test and diagnostic purposes

The output latches of P1.6 and P1.7 must be set to logic 1 in order to enable SIO1.

The 8XC552 on-chip I^2C logic provides a serial interface that meets the I^2C bus specification and supports all transfer modes (other than the low-speed mode) from and to the I^2C bus. The SIO1 logic handles bytes transfer autonomously. It also keeps track of serial transfers, and a status register (S1STA) reflects the status of SIO1 and the I^2C bus.

The CPU interfaces to the I²C logic via the following four special function registers: S1CON (SIO1 control register), S1STA (SIO1 status register), S1DAT (SIO1 data register), and S1ADR (SIO1 slave address register). The SIO1 logic interfaces to the external I²C bus via two port 1 pins: P1.6/SCL (serial clock line) and P1.7/SDA (serial data line).

A typical I²C bus configuration is shown in Figure 10, and Figure 11 shows how a data transfer is accomplished on the bus. Depending on the state of the direction bit (R/W), two types of data transfers are possible on the I²C bus:

1. Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte.
2. Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a "not acknowledge" is returned.

The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the I²C bus will not be released.

Modes of Operation: The on-chip SIO1 logic may operate in the following four modes:

1. Master Transmitter Mode:

   Serial data output through P1.7/SDA while P1.6/SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the data direction bit. In this case the data direction bit (R/W) will be logic 0, and we say that a "W" is transmitted. Thus the first byte transmitted is SLA+W. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.
2. Master Receiver Mode:

   The first byte transmitted contains the slave address of the transmitting device (7 bits) and the data direction bit. In this case the data direction bit (R/W) will be logic 1, and we say that an "R" is transmitted. Thus the first byte transmitted is SLA+R. Serial data is received via P1.7/SDA while P1.6/SCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are output to indicate the beginning and end of a serial transfer.
3. Slave Receiver Mode:

   Serial data and the serial clock are received through P1.7/SDA and P1.6/SCL. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and direction bit.
4. Slave Transmitter Mode:

   The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will indicate that the transfer direction is reversed. Serial data is transmitted via P1.7/SDA while the serial clock is input through P1.6/SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer.

In a given application, SIO1 may operate as a master and as a slave. In the slave mode, the SIO1 hardware looks for its own slave address and the general call address. If one of these addresses is detected, an interrupt is requested. When the microcontroller wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave action is not interrupted. If bus arbitration is lost in the master mode, SIO1 switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

SIO1 Implementation and Operation: Figure 12 shows how the on-chip I²C bus interface is implemented, and the following text describes the individual blocks.

Input Filters and Output Stages

The input filters have I²C compatible input levels. If the input voltage is less than 1.5V, the input logic level is interpreted as 0; if the input voltage is greater than 3.0V, the input logic level is interpreted as 1. Input signals are synchronized with the internal clock (fosc/4), and spikes shorter than three oscillator periods are filtered out.

The output stages consist of open drain transistors that can sink 3mA at $V_{OUT} < 0.4V$. These open drain outputs do not have clamping diodes to V_{DD}. Thus, if the device is connected to the I²C bus and V_{DD} is switched off, the I²C bus is not affected.

Address Register, S1ADR

This 8-bit special function register may be loaded with the 7-bit slave address (7 most significant bits) to which SIO1 will respond when programmed as a slave transmitter or receiver. The LSB (GC) is used to enable general call address (00H) recognition.

Comparator

The comparator compares the received 7-bit slave address with its own slave address (7 most significant bits in S1ADR). It also compares the first received 8-bit byte with the general call address (00H). If an equality is found, the appropriate status bits are set and an interrupt is requested.

Shift Register, S1DAT

This 8-bit special function register contains a byte of serial data to be transmitted or a byte which has just been received. Data in S1DAT is always shifted from right to left; the first bit to be transmitted is the MSB (bit 7) and, after a byte has been received, the first bit of received data is located at the MSB of S1DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; S1DAT always contains the last byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in S1DAT.

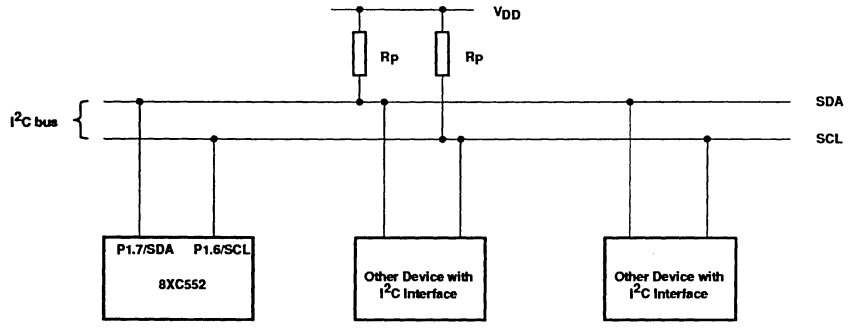


Figure 10. Typical I²C Bus Configuration

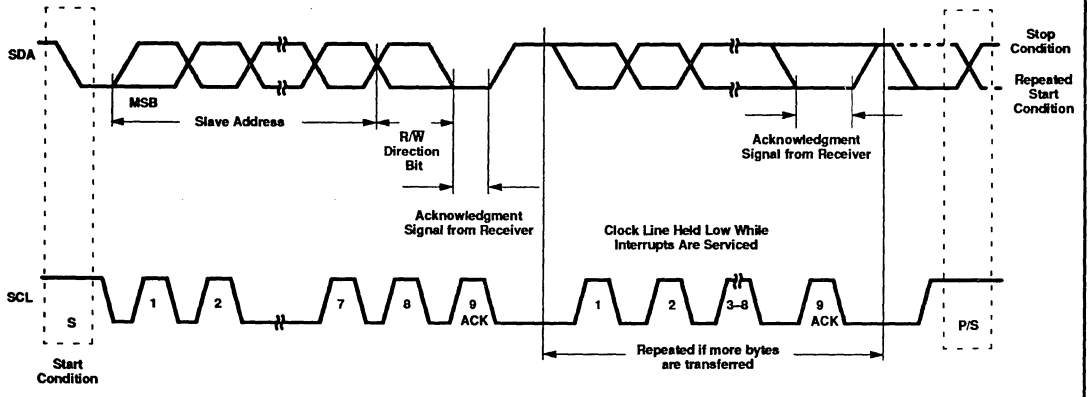


Figure 11. Data Transfer on the I²C Bus

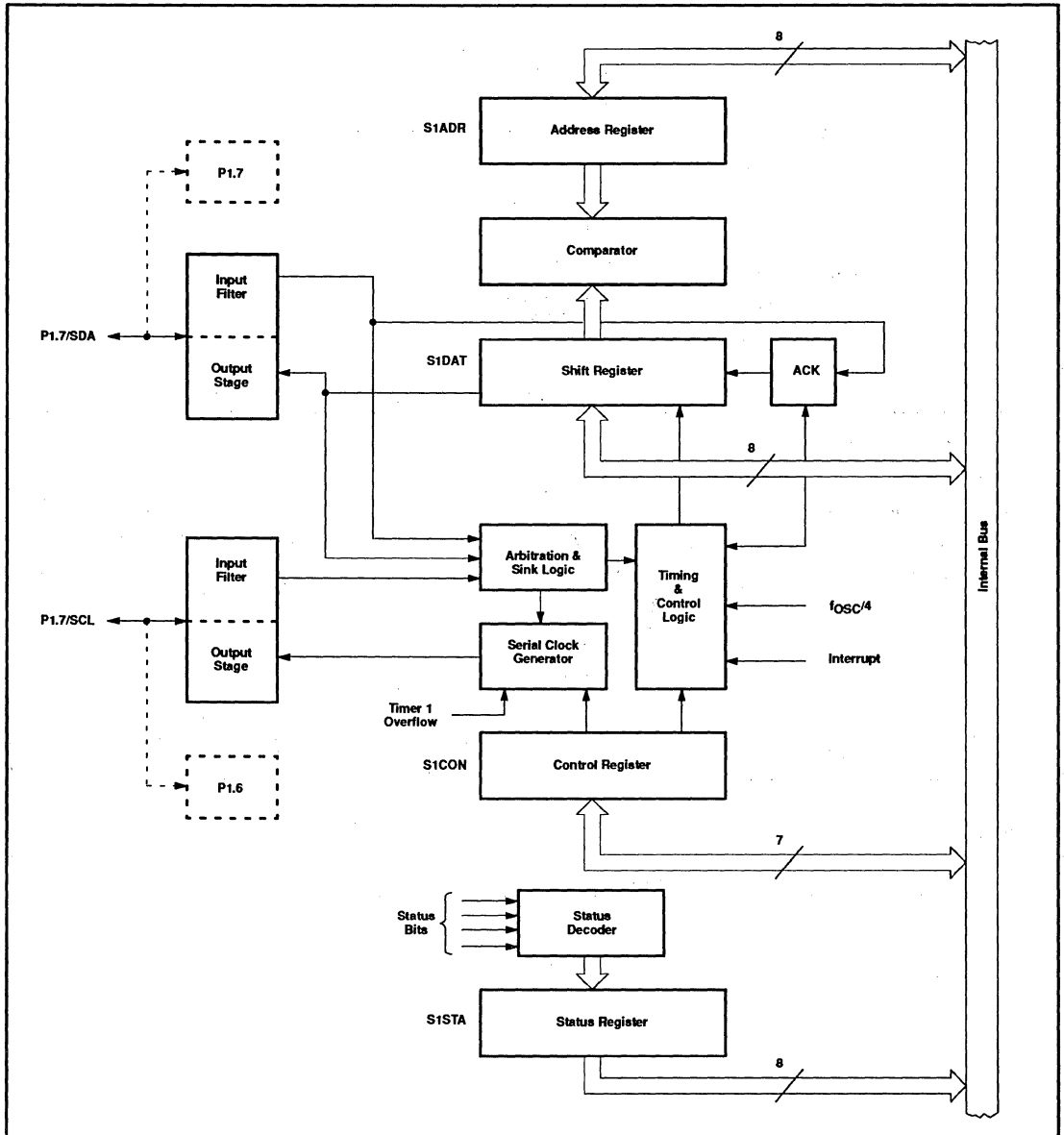
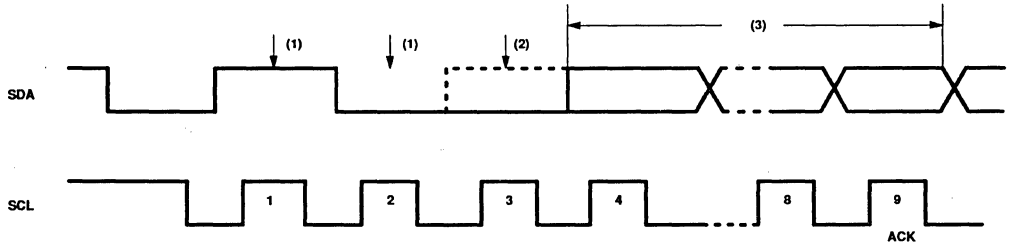
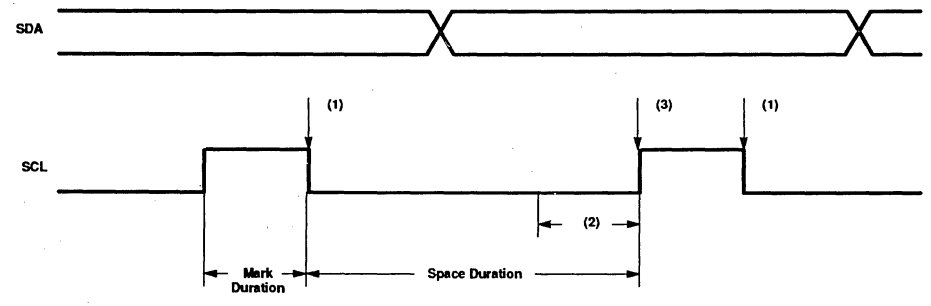


Figure 12. I²C Bus Serial Interface Block Diagram



1. Another device transmits identical serial data.
2. Another device overrules a logic 1 (dotted line) transmitted by SIO1 (master) by pulling the SDA line low. Arbitration is lost, and SIO1 enters the slave receiver mode.
3. SIO1 is in the slave receiver mode but still generates clock pulses until the current byte has been transmitted. SIO1 will not generate clock pulses for the next byte. Data on SDA originates from the new master once it has won arbitration.

Figure 13. Arbitration Procedure



1. Another service pulls the SCL line low before the SIO1 "mark" duration is complete. The serial clock generator is immediately reset and commences with the "space" duration by pulling SCL low.
2. Another device still pulls the SCL line low after SIO1 releases SCL. The serial clock generator is forced into the wait state until the SCL line is released.
3. The SCL line is released, and the serial clock generator commences with the mark duration.

Figure 14. Serial Clock Synchronization

Arbitration and Synchronization Logic

In the master transmitter mode, the arbitration logic checks that every transmitted logic 1 actually appears as a logic 1 on the I^2C bus. If another device on the bus overrules a logic 1 and pulls the SDA line low, arbitration is lost, and SIO1 immediately changes from master transmitter to slave receiver. SIO1 will continue to output clock pulses (on SCL) until transmission of the current serial byte is complete.

Arbitration may also be lost in the master receiver mode. Loss of arbitration in this mode can only occur while SIO1 is returning a "not acknowledge: (logic 1) to the bus. Arbitration is lost when another device on the bus pulls this signal LOW. Since this can occur only at the end of a serial byte, SIO1 generates no further clock pulses. Figure 13 shows the arbitration procedure.

The synchronization logic will synchronize the serial clock generator with the clock pulses on the SCL line from another device. If two or more master devices generate clock pulses, the "mark" duration is determined by the device that generates the shortest "marks," and the "space" duration is determined by the device that generates the longest "spaces." Figure 14 shows the synchronization procedure.

A slave may stretch the space duration to slow down the bus master. The space duration may also be stretched for handshaking purposes. This can be done after each bit or after a complete byte transfer. SIO1 will stretch the SCL space duration after a byte has been transmitted or received and the acknowledge bit has been transferred. The serial interrupt flag (SI) is set, and the stretching continues until the serial interrupt flag is cleared.

Serial Clock Generator

This programmable clock pulse generator provides the SCL clock pulses when SIO1 is in the master transmitter or master receiver mode. It is switched off when SIO1 is in a slave mode. The programmable output clock frequencies are: fosc/120, fosc/9600, and the Timer 1 overflow rate divided by eight. The output clock pulses have a 50% duty cycle unless the clock generator is synchronized with other SCL clock sources as described above.

Timing and Control

The timing and control logic generates the timing and control signals for serial byte handling. This logic block provides the shift pulses for S1DAT, enables the comparator, generates and detects start and stop conditions, receives and transmits acknowledge bits, controls the master and slave modes, contains interrupt request logic, and monitors the I^2C bus status.

Control Register, S1CON

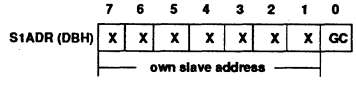
This 7-bit special function register is used by the microcontroller to control the following SIO1 functions: start and restart of a serial transfer, termination of a serial transfer, bit rate, address recognition, and acknowledgment.

Status Decoder and Status Register

The status decoder takes all of the internal status bits and compresses them into a 5-bit code. This code is unique for each I^2C bus status. The 5-bit code may be used to generate vector addresses for fast processing of the various service routines. Each service routine processes a particular bus status. There are 26 possible bus states if all four modes of SIO1 are used. The 5-bit status code is latched into the five most significant bits of the status register when the serial interrupt flag is set (by hardware) and remains stable until the interrupt flag is cleared by software. The three least significant bits of the status register are always zero. If the status code is used as a vector to service routines, then the routines are displaced by eight address locations. Eight bytes of code is sufficient for most of the service routines (see the software example in this section).

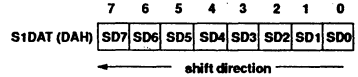
The Four SIO1 Special Function Registers: The microcontroller interfaces to SIO1 via four special function registers. These four SFRs (S1ADR, S1DAT, S1CON, and S1STA) are described individually in the following sections.

The Address Register, S1ADR: The CPU can read from and write to this 8-bit, directly addressable SFR. S1ADR is not affected by the SIO1 hardware. The contents of this register are irrelevant when SIO1 is in a master mode. In the slave modes, the seven most significant bits must be loaded with the microcontroller's own slave address, and, if the least significant bit is set, the general call address (00H) is recognized; otherwise it is ignored.



The most significant bit corresponds to the first bit received from the I^2C bus after a start condition. A logic 1 in S1ADR corresponds to a high level on the I^2C bus, and a logic 0 corresponds to a low level on the bus.

The Data Register, S1DAT: S1DAT contains a byte of serial data to be transmitted or a byte which has just been received. The CPU can read from and write to this 8-bit, directly addressable SFR while it is not in the process of shifting a byte. This occurs when SIO1 is in a defined state and the serial interrupt flag is set. Data in S1DAT remains stable as long as SI is set. Data in S1DAT is always shifted from right to left: the first bit to be transmitted is the MSB (bit 7), and, after a byte has been received, the first bit of received data is located at the MSB of S1DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; S1DAT always contains the last data byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in S1DAT.



SD7 - SD0:

Eight bits to be transmitted or just received. A logic 1 in S1DAT corresponds to a high level on the I^2C bus, and a logic 0 corresponds to a low level on the bus. Serial data shifts through S1DAT from right to left. Figure 15 shows how data in S1DAT is serially transferred to and from the SDA line.

S1DAT and the ACK flag form a 9-bit shift register which shifts in or shifts out an 8-bit byte, followed by an acknowledge bit. The ACK flag is controlled by the SIO1 hardware and cannot be accessed by the CPU. Serial data is shifted through the ACK flag into S1DAT on the rising edges of serial clock pulses on the SCL line. When a byte has been shifted into S1DAT, the serial data is available in S1DAT, and the acknowledge bit is returned by the control logic during the ninth clock pulse. Serial data is shifted out from S1DAT via a buffer (BSD7) on the falling edges of clock pulses on the SCL line.

When the CPU writes to S1DAT, BSD7 is loaded with the content of S1DAT.7, which is the first bit to be transmitted to the SDA line (see Figure 16). After nine serial clock pulses, the eight bits in S1DAT will have been transmitted to the SDA line, and the acknowledge bit will be present in ACK. Note that the eight transmitted bits are shifted back into S1DAT.

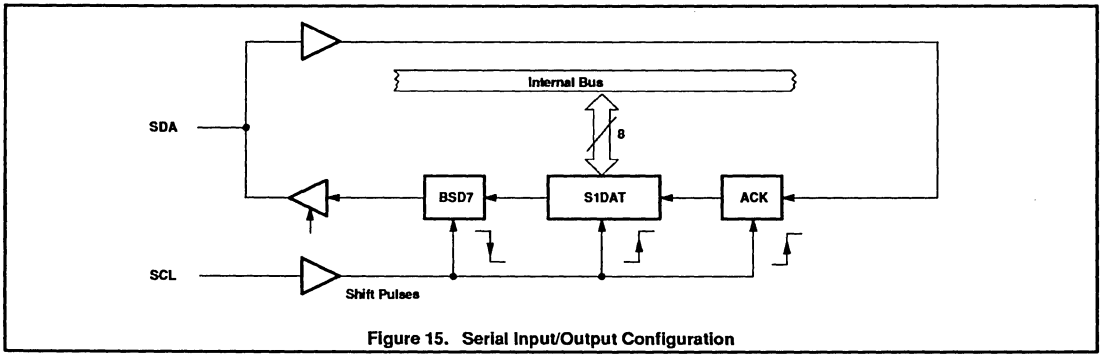


Figure 15. Serial Input/Output Configuration

The Control Register, S1CON: The CPU can read from and write to this 8-bit, directly addressable SFR. Two bits are affected by the SIO1 hardware: the SI bit is set when a serial interrupt is requested, and the STO bit is cleared when a STOP condition is present on the I²C bus. The STO bit is also cleared when ENS1 = "0".

	7	6	5	4	3	2	1	0
S1CON (D8H)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

ENS1, the SIO1 Enable Bit

ENS1 = "0": When ENS1 is "0", the SDA and SCL outputs are in a high impedance state. SDA and SCL input signals are ignored, SIO1 is in the "not addressed" slave state, and the STO bit in S1CON is forced to "0". No other bits are affected. P1.6 and P1.7 may be used as open drain I/O ports.

ENS1 = "1": When ENS1 is "1", SIO1 is enabled. The P1.6 and P1.7 port latches must be set to logic 1.

ENS1 should not be used to temporarily release SIO1 from the I2C bus since, when ENS1 is reset, the I2C bus status is lost. The AA flag should be used instead (see description of the AA flag in the following text).

In the following text, it is assumed that ENS1 = "1".

STA, the START Flag

STA = "1": When the STA bit is set to enter a master mode, the SIO1 hardware checks the status of the I2C bus and generates a START condition if the bus is free. If the bus is not free, then SIO1 waits for a STOP condition (which will free the bus) and generates a START condition after a delay of a half clock period of the internal serial clock generator.

If STA is set while SIO1 is already in a master mode and one or more bytes are transmitted or received, SIO1 transmits a repeated START condition. STA may be set at any time. STA may also be set when SIO1 is an addressed slave.

STA = "0": When the STA bit is reset, no START condition or repeated START condition will be generated.

STO, the STOP Flag

STO = "1": When the STO bit is set while SIO1 is in a master mode, a STOP condition is transmitted to the I²C bus. When the STOP condition is detected on the bus, the SIO1 hardware clears the STO flag. In a slave mode, the STO flag may be set to recover from an error condition. In this case, no STOP condition is transmitted to the I²C bus. However, the SIO1 hardware behaves as if a STOP condition has been received and switches to the defined "not addressed" slave receiver mode. The STO flag is automatically cleared by hardware.

If the STA and STO bits are both set, the a STOP condition is transmitted to the I²C bus if SIO1 is in a master mode (in a slave mode, SIO1 generates an internal STOP condition which is not transmitted). SIO1 then transmits a START condition.

STO = "0": When the STO bit is reset, no STOP condition will be generated.

SI, the Serial Interrupt Flag

SI = "1": When the SI flag is set, then, if the EA and ES1 (interrupt enable register) bits are also set, a serial interrupt is requested. SI is set by hardware when one of 25 of the 26 possible SIO1 states is entered. The only state that does not cause SI to be set is state F8H, which indicates that no relevant state information is available.

While SI is set, the low period of the serial clock on the SCL line is stretched, and the serial transfer is suspended. A high level on the SCL line is unaffected by the serial interrupt flag. SI must be reset by software.

SI = "0": When the SI flag is reset, no serial interrupt is requested, and there is no stretching of the serial clock on the SCL line.

AA, the Assert Acknowledge Flag

AA = "1": If the AA flag is set, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when:
- The "own slave address" has been received
- The general call address has been received while the general call bit (GC) in S1ADR is set
- A data byte has been received while SIO1 is in the master receiver mode
- A data byte has been received while SIO1 is in the addressed slave receiver mode

AA = "0": if the AA flag is reset, a not acknowledge (high level to SDA) will be returned during the acknowledge clock pulse on SCL when:
- A data has been received while SIO1 is in the master receiver mode
- A data byte has been received while SIO1 is in the addressed slave receiver mode

When WIO1 is in the addressed slave transmitter mode, state C8H will be entered after the last serial is transmitted (see Figure 20). When SI is cleared, SIO1 leaves state C8H, enters the not addressed slave receiver mode, and the SDA line remains at a high level. In state C8H, the AA flag can be set again for future address recognition.

When SIO1 is in the not addressed slave mode, its own slave address and the general call address are ignored. Consequently, no acknowledge is returned, and a serial interrupt is not requested. Thus, SIO1 can be temporarily released from the I²C bus while the bus status is monitored. While SIO1 is released from the bus, START and STOP conditions are detected, and serial data is shifted in. Address recognition can be

resumed at any time by setting the AA flag. If the AA flag is set when the part's own slave address or the general call address has been partly received, the address will be recognized at the end of the byte transmission.

CR0, CR1, and CR2, the Clock Rate Bits

These three bits determine the serial clock frequency when SIO1 is in a master mode. The various serial rates are shown in Table 2.

A 12.5kHz bit rate may be used by devices that interface to the I^2C bus via standard I/O port lines which are software driven and slow. 100kHz is usually the maximum bit rate and can be derived from a 16MHz, 12MHz, or a 6MHz oscillator. A variable bit rate (0.5kHz to 62.5kHz) may also be used if Timer 1 is not required for any other purpose while SIO1 is in a master mode.

The frequencies shown in Table 2 are unimportant when SIO1 is in a slave mode. In the slave modes, SIO1 will automatically synchronize with any clock frequency up to 100kHz.

The Status Register, S1STA: S1STA is an 8-bit read-only special function register. The three least significant bits are always zero. The five most significant bits contain the status code. There are 26 possible status codes. When S1STA contains F8H, no relevant state information is available and no serial interrupt is requested. All other S1STA values correspond to defined SIO1 states. When each of these states is entered, a serial interrupt is requested (SI = "1"). A valid status code is present in S1STA one machine cycle after SI is set by hardware and is still present one machine cycle after SI has been reset by software.

More Information on SIO1 Operating Modes: The four operating modes are:

- Master Transmitter
- Master Receiver
- Slave Receiver
- Slave Transmitter

Data transfers in each mode of operation are shown in Figures 17–37. These figures contain the following abbreviations:

Abbreviation	Explanation
S	Start condition
SLA	7-bit slave address
R	Read bit (high level at SDA)
W	Write bit (low level at SDA)
A	Acknowledge bit (low level at SDA)
$\overline{A}$	Not acknowledge bit (high level at SDA)
Data	8-bit data byte
P	Stop condition

In Figures 17-37, circles are used to indicate when the serial interrupt flag is set. The numbers in the circles show the status code held in the S1STA register. At these points, a service routine must be executed to continue or complete the serial transfer. These service routines are not critical since the serial transfer is suspended until the serial interrupt flag is cleared by software.

When a serial interrupt routine is entered, the status code in S1STA is used to branch to the appropriate service routine. For each status code, the required software action and details of the following serial transfer are given in Tables 15-18.

Master Transmitter Mode: In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see Figure 17). Before the master transmitter mode can be entered, S1CON must be initialized as follows:

	7	6	5	4	3	2	1	0
S1CON (D8H)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0
	bit rate	1	0	0	0	X	bit rate	

CR0, CR1, and CR2 define the serial bit rate. ENS1 must be set to logic 1 to enable SIO1. If the AA bit is reset, SIO1 will not acknowledge its own slave address or the general call address in the event of another device becoming master of the bus. In other words, if AA is reset, SIO0 cannot enter a slave mode. STA, STO, and SI must be reset.

The master transmitter mode may now be entered by setting the STA bit using the SETB instruction. The SIO1 logic will now test the I^2C bus and generate a start condition as soon as the bus becomes free. When a START condition is transmitted, the serial interrupt flag (SI) is set, and the status code in the status register (S1STA) will be 08H. This status code must be used to vector to an interrupt service routine that loads S1DAT with the slave address and the data direction bit (SLA+W). The SI bit in S1CON must then be reset before the serial transfer can continue.

When the slave address and the direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in S1STA are possible. There are 18H, 20H, or 38H for the master mode and also 68H, 78H, or B0H if the slave mode was enabled (AA = logic 1). The appropriate action to be taken for each of these status codes is detailed in Table 3. After a repeated start condition (state 10H). SIO1 may switch to the master receiver mode by loading S1DAT with SLA+R).

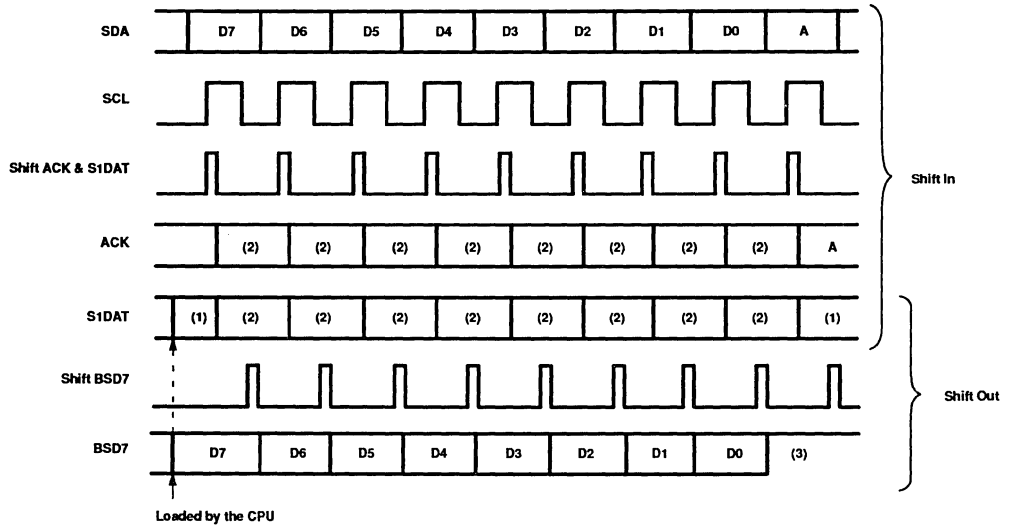


Figure 16. Shift-in and Shift-out Timing

Table 2. Serial Clock Rates

CR2	CR1	CR0	BIT FREQUENCY (kHz) AT f_{OSC} 6MHz	12MHz	16MHz	f_{OSC} DIVIDED BY
0	0	0	23	47	63	256
0	0	1	27	54	71	224
0	1	0	31	63	83	192
0	1	1	37	75	100	160
1	0	0	6.25	12.5	17	960
1	0	1	50	100	–	120
1	1	0	100	–	–	60
1	1	1	0.25 < 62.5	0.5 < 62.5	0.67 < 56	96 × (256 – reload value Timer 1) (Reload value range: 0 – 254 in mode 2)

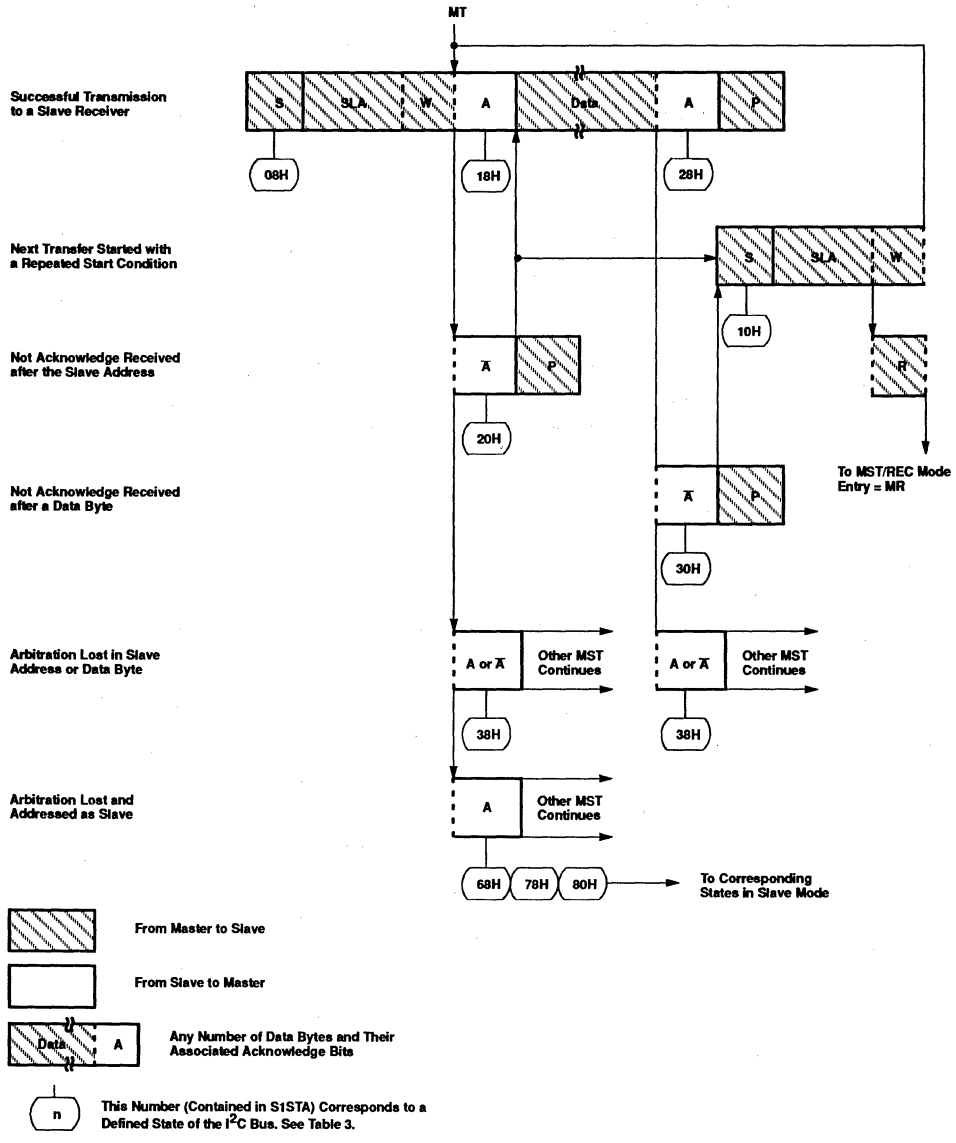


Figure 17. Format and States in the Master Transmitter Mode

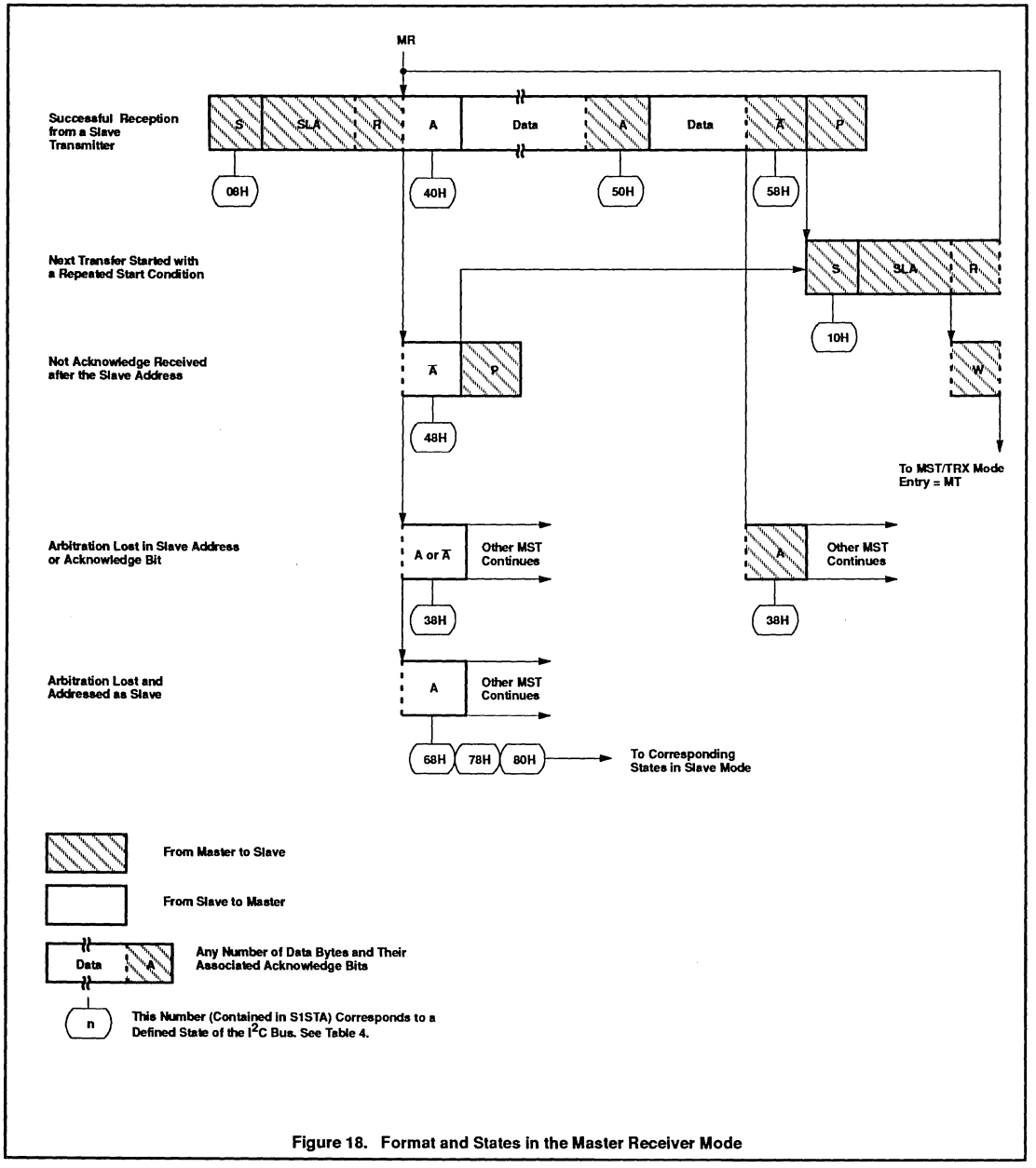


Figure 18. Format and States in the Master Receiver Mode

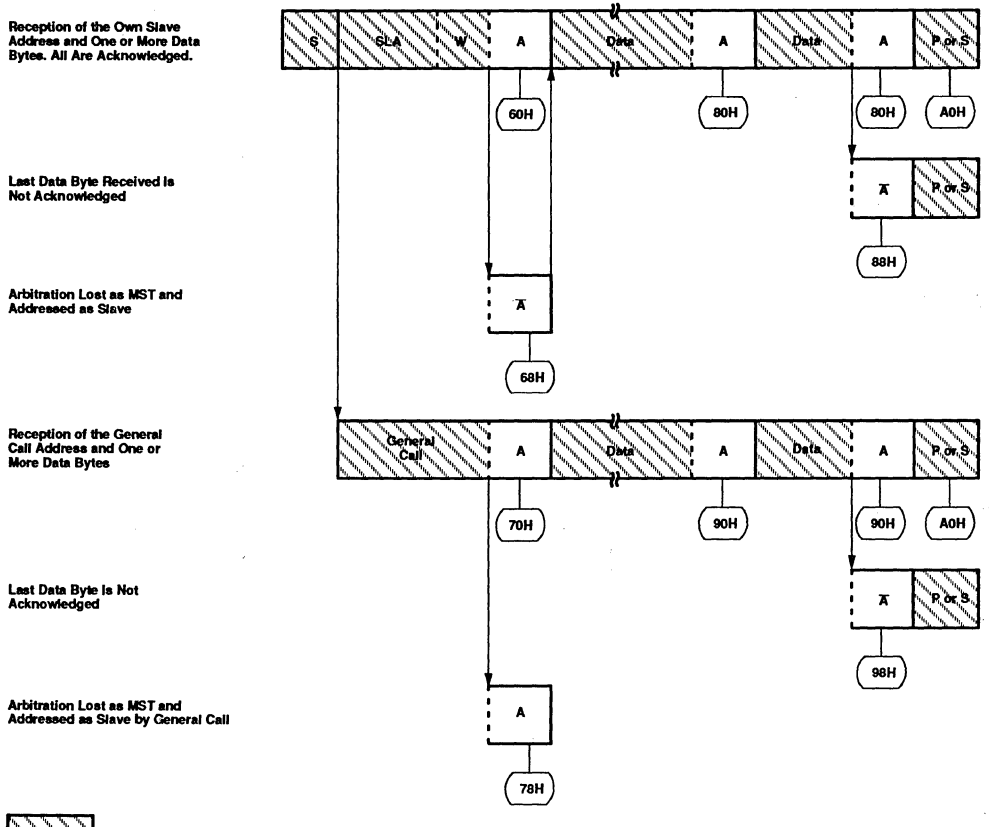


Figure 19. Format and States in the Slave Receiver Mode

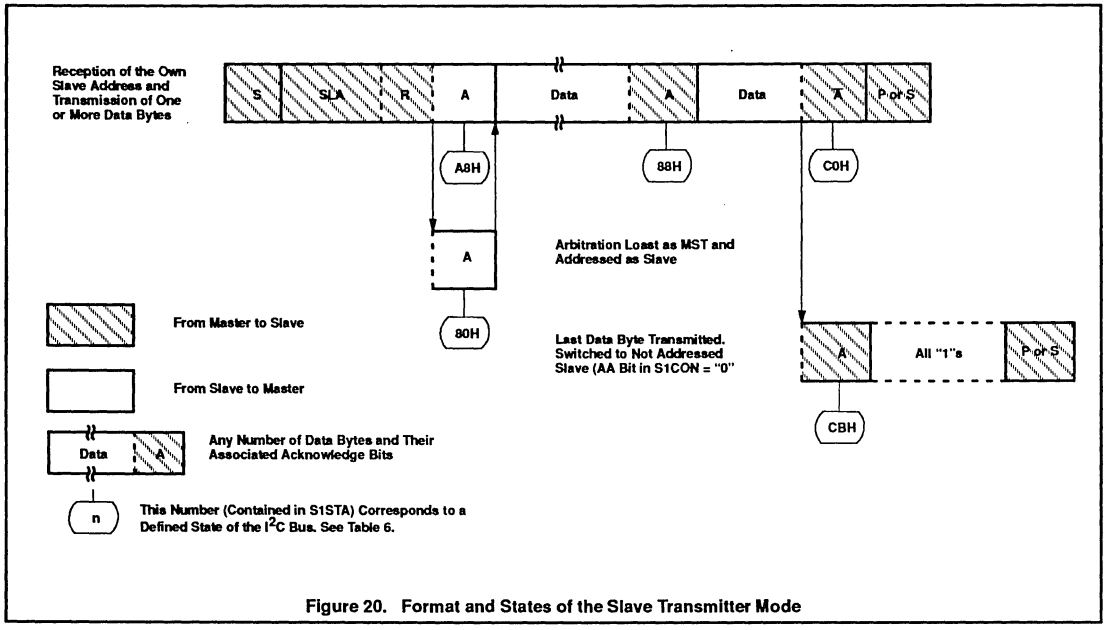


Figure 20. Format and States of the Slave Transmitter Mode

Master Receiver Mode: In the master receiver mode, a number of data bytes are received from a slave transmitter (see Figure 18). The transfer is initialized as in the master transmitter mode. When the start condition has been transmitted, the interrupt service routine must load S1DAT with the 7-bit slave address and the data direction bit (SLA+R). The SI bit in S1CON must then be cleared before the serial transfer can continue.

When the slave address and the data direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in S1STA are possible. These are 40H, 48H, or 38H for the master mode and also 68H, 78H, or B0H if the slave mode was enabled (AA = logic 1). The appropriate action to be taken for each of these status codes is detailed in Table 4. ENS1, CR1, and CR0 are not affected by the serial transfer and are not referred to in Table 4. After a repeated start condition (state 10H), SIO1 may switch to the master transmitter mode by loading S1DAT with SLA+W.

Slave Receiver Mode: In the slave receiver mode, a number of data bytes are received from a master transmitter (see Figure 19). To initiate the slave receiver mode, S1ADR and S1CON must be loaded as follows:

	7	6	5	4	3	2	1	0
S1ADR (DBH)	X	X	X	X	X	X	X	GC

own slave address

The upper 7 bits are the address to which SIO1 will respond when addressed by a master. If the LSB (GC) is set, SIO1 will respond to the general call address (00H); otherwise it ignores the general call address.

	7	6	5	4	3	2	1	0
S1CON (D8H)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0
	X	1	0	0	0	1	X	X

CR0, CR1, and CR12 do not affect SIO1 in the slave mode. ENS1 must be set to logic 1 to enable SIO1. The AA bit must be set to enable SIO1 to acknowledge its own slave address or the general call address. STA, STO, and SI must be reset.

When S1ADR and S1CON have been initialized, SIO1 waits until it is addressed by its own slave address followed by the data direction bit which must be "0" (W) for SIO1 to operate in the slave receiver mode. After its own slave address and the W bit have been received, the serial interrupt flag (I) is set and a valid status code can be read from S1STA. This status code is used to vector to an interrupt service routine, and the appropriate action to be taken for each of these status codes is detailed in Table 5. The slave receiver mode may also be entered if arbitration is lost while SIO1 is in the master mode (see status 68H and 78H).

If the AA bit is reset during a transfer, SIO1 will return a not acknowledge (logic 1) to SDA after the next received data byte. While AA is reset, SIO1 does not respond to its own slave address or a general call address. However, the I²C bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate SIO1 from the I²C bus.

Table 3. Master Transmitter Mode

STATUS CODE (S1STA)	STATUS OF THE I²C BUS AND SIO1 HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY SIO1 HARDWARE
		TO/FROM S1DAT	TO S1CON				
			STA	STO	SI	AA	
08H	A START condition has been transmitted	Load SLA+W	X	0	0	X	SLA+W will be transmitted; ACK bit will be received
10H	A repeated START condition has been transmitted	Load SLA+W or Load SLA+R	X X	0 0	0 0	X X	As above SLA+W will be transmitted; SIO1 will be switched to MST/REC mode
18H	SLA+W has been transmitted; ACK has been received	Load data byte or no S1DAT action or no S1DAT action or no S1DAT action	0 1 0 1	0 0 1 1	0 0 0 0	X X X X	Data byte will be transmitted; ACK bit will be received Repeated START will be transmitted; STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
20H	SLA+W has been transmitted; NOT ACK has been received	Load data byte or no S1DAT action or no S1DAT action or no S1DAT action	0 1 0 1	0 0 1 1	0 0 0 0	X X X X	Data byte will be transmitted; ACK bit will be received Repeated START will be transmitted; STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
28H	Data byte in S1DAT has been transmitted; ACK has been received	Load data byte or no S1DAT action or no S1DAT action or no S1DAT action	0 1 0 1	0 0 1 1	0 0 0 0	X X X X	Data byte will be transmitted; ACK bit will be received Repeated START will be transmitted; STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
30H	Data byte in S1DAT has been transmitted; NOT ACK has been received	Load data byte or no S1DAT action or no S1DAT action or no S1DAT action	0 1 0 1	0 0 1 1	0 0 0 0	X X X X	Data byte will be transmitted; ACK bit will be received Repeated START will be transmitted; STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
38H	Arbitration lost in SLA+R/$\overline{W}$ or Data bytes	No S1DAT action or No S1DAT action	0 1	0 0	0 0	X X	I²C bus will be released; not addressed slave will be entered A START condition will be transmitted when the bus becomes free

Table 4. Master Receiver Mode

STATUS CODE (S1STA)	STATUS OF THE I²C BUS AND SIO1 HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY SIO1 HARDWARE
		TO/FROM S1DAT	TO S1CON				
			STA	STO	SI	AA	
08H	A START condition has been transmitted	Load SLA+R	X	0	0	X	SLA+R will be transmitted; ACK bit will be received
10H	A repeated START condition has been transmitted	Load SLA+R or Load SLA+W	X X	0 0	0 0	X X	As above SLA+W will be transmitted; SIO1 will be switched to MST/TRX mode
38H	Arbitration lost in NOT ACK bit	No S1DAT action or No S1DAT action	0 1	0 0	0 0	X X	I²C bus will be released; SIO1 will enter a slave mode A START condition will be transmitted when the bus becomes free
40H	SLA+R has been transmitted; ACK has been received	No S1DAT action or no S1DAT action	0 0	0 0	0 0	0 1	Data byte will be received; NOT ACK bit will be returned Data byte will be received; ACK bit will be returned
48H	SLA+R has been transmitted; NOT ACK has been received	No S1DAT action or no S1DAT action or no S1DAT action	1 0 1	0 1 1	0 0 0	X X X	Repeated START condition will be transmitted STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset
50H	Data byte has been received; ACK has been returned	Read data byte or read data byte	0 0	0 0	0 0	0 1	Data byte will be received; NOT ACK bit will be returned Data byte will be received; ACK bit will be returned
58H	Data byte has been received; NOT ACK has been returned	Read data byte or read data byte or read data byte	1 0 1	0 1 1	0 0 0	X X X	Repeated START condition will be transmitted STOP condition will be transmitted; STO flag will be reset STOP condition followed by a START condition will be transmitted; STO flag will be reset

Table 5. Slave Receiver Mode

STATUS CODE (S1STA)	STATUS OF THE I²C BUS AND SIO1 HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY SIO1 HARDWARE
		TO/FROM S1DAT	TO S1CON				
			STA	STO	SI	AA	
60H	Own SLA+W has been received; ACK has been returned	No S1DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned
		no S1DAT action	X	0	0	1	Data byte will be received and ACK will be returned
68H	Arbitration lost in SLA+R/W as master; Own SLA+W has been received, ACK returned	No S1DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned
		no S1DAT action	X	0	0	1	Data byte will be received and ACK will be returned
70H	General call address (00H) has been received; ACK has been returned	No S1DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned
		no S1DAT action	X	0	0	1	Data byte will be received and ACK will be returned
78H	Arbitration lost in SLA+R/W as master; General call address has been received, ACK has been returned	No S1DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned
		no S1DAT action	X	0	0	1	Data byte will be received and ACK will be returned
80H	Previously addressed with own SLV address; DATA has been received; ACK has been returned	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned
		read data byte	X	0	0	1	Data byte will be received and ACK will be returned
88H	Previously addressed with own SLA; DATA byte has been received; NOT ACK has been returned	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address
		read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1
		read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free
		read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.
90H	Previously addressed with General Call; DATA byte has been received; ACK has been returned	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned
		read data byte	X	0	0	1	Data byte will be received and ACK will be returned
98H	Previously addressed with General Call; DATA byte has been received; NOT ACK has been returned	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address
		read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1
		read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free
		read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.

Table 5. Slave Receiver Mode (Continued)

STATUS CODE (S1STA)	STATUS OF THE I²C BUS AND SIO1 HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY SIO1 HARDWARE
		TO/FROM S1DAT	TO S1CON				
			STA	STO	SI	AA	
A0H	A STOP condition or repeated START condition has been received while still addressed as SLV/REC or SLV/TRX	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address
		read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1
		read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free
		read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.

Table 6. Slave Transmitter Mode

STATUS CODE (S1STA)	STATUS OF THE I²C BUS AND SIO1 HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY SIO1 HARDWARE
		TO/FROM S1DAT	TO S1CON				
			STA	STO	SI	AA	
A8H	Own SLA+R has been received; ACK has been returned	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received
		load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received
B0H	Arbitration lost in SLA+R/$\overline{W}$ as master; Own SLA+R has been received, ACK has been returned	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received
		load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received
B8H	Data byte in S1DAT has been transmitted; ACK has been received	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received
		load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received
C0H	Data byte in S1DAT has been transmitted; NOT ACK has been received	No S1DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address
		no S1DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1
		no S1DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free
		no S1DAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.
C8H	Last data byte in S1DAT has been transmitted (AA = 0); ACK has been received	No S1DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address
		no S1DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1
		no S1DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General call address. A START condition will be transmitted when the bus becomes free
		no S1DAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General call address will be recognized if S1ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.

Slave Transmitter Mode: In the slave transmitter mode, a number of data bytes are transmitted to a master receiver (see Figure 20). Data transfer is initialized as in the slave receiver mode. When S1ADR and S1CON have been initialized, SIO1 waits until it is addressed by its own slave address followed by the data direction bit which must be "1" (R) for SIO1 to operate in the slave transmitter mode. After its own slave address and the R bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from S1STA. This status code is used to vector to an interrupt service routine, and the appropriate action to be taken for each of these status codes is detailed in Table 6. The slave transmitter mode may also be entered if arbitration is lost while SIO1 is in the master mode (see state B0H).

If the AA bit is reset during a transfer, SIO1 will transmit the last byte of the transfer and enter state C0H or C8H. SIO1 is switched to the not addressed slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all 1s as serial data. While AA is reset, SIO1 does not respond to its own slave address or a general call address. However, the I^2C bus is still monitored, and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate SIO1 from the I^2C bus.

Miscellaneous States: There are two S1STA codes that do not correspond to a defined SIO1 hardware state (see Table 7). These are discussed below.

S1STA = F8H:

This status code indicates that no relevant information is available because the serial interrupt flag, SI, is not yet set. This occurs between other states and when SIO1 is not involved in a serial transfer.

S1STA = 00H:

This status code indicates that a bus error has occurred during an SIO1 serial transfer. A bus error is caused when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. A bus error may also be caused when external interference disturbs the internal SIO1 signals. When a bus error occurs, SI is set. To recover from a bus error, the STO flag must be set and SI must be cleared. This causes SIO1 to enter the "not addressed" slave mode (a defined state) and to clear the STO flag (no other bits in S1CON are affected). The SDA and SCL lines are released (a STOP condition is not transmitted).

Some Special Cases: The SIO1 hardware has facilities to handle the following special cases that may occur during a serial transfer:

Simultaneous Repeated START Conditions from Two Masters

A repeated START condition may be generated in the master transmitter or master receiver modes. A special case occurs if another master simultaneously generates a repeated START condition (see Figure 21). Until this occurs, arbitration is not lost by either master since they were both transmitting the same data.

If the SIO1 hardware detects a repeated START condition on the I^2C bus before generating a repeated START condition itself, it will release the bus, and no interrupt request is generated. If another master frees the bus by generating a STOP condition, SIO1 will transmit a normal START condition (state 08H), and a retry of the total serial data transfer can commence.

Data Transfer After Loss of Arbitration

Arbitration may be lost in the master transmitter and master receiver modes (see Figure 13). Loss of arbitration is indicated by the following states in S1STA; 38H, 68H, 78H, and B0H (see Figures 17 and 18).

If the STA flag in S1CON is set by the routines which service these states, then, if the bus is free again, a START condition (state 08H) is transmitted without intervention by the CPU, and a retry of the total serial transfer can commence.

Forced Access to the I^2C Bus

In some applications, it may be possible for an uncontrolled source to cause a bus hang-up. In such situations, the problem may be caused by interference, temporary interruption of the bus or a temporary short-circuit between SDA and SCL.

If an uncontrolled source generates a superfluous START or masks a STOP condition, then the I^2C bus stays busy indefinitely. If the STA flag is set and bus access is not obtained within a reasonable amount of time, then a forced access to the I^2C bus is possible. This is achieved by setting the STO flag while the STA flag is still set. No STOP condition is transmitted. The SIO1 hardware behaves as if a STOP condition was received and is able to transmit a START condition. The STO flag is cleared by hardware (see Figure 22).

I^2C Bus Obstructed by a Low Level on SCL or SDA

An I^2C bus hang-up occurs if SDA or SCL is pulled LOW by an uncontrolled source. If the SCL line is obstructed (pulled LOW) by a device on the bus, no further serial transfer is possible, and the SIO1 hardware cannot resolve this type of problem. When this occurs, the problem must be resolved by the device that is pulling the SCL bus line LOW.

If the SDA line is obstructed by another device on the bus (e.g., a slave device out of bit synchronization), the problem can be solved by transmitting additional clock pulses on the SCL line (see Figure 23). The SIO1 hardware transmits additional clock pulses when the STA flag is set, but no START condition can be generated because the SDA line is pulled LOW while the I^2C bus is considered free. The SIO1 hardware attempts to generate a START condition after every two additional clock pulses on the SCL line. When the SDA line is eventually released, a normal START condition is transmitted, state 08H is entered, and the serial transfer continues.

If a forced bus access occurs or a repeated START condition is transmitted while SDA is obstructed (pulled LOW), the SIO1 hardware performs the same action as described above. In each case, state 08H is entered after a successful START condition is transmitted and normal serial transfer continues. Note that the CPU is not involved in solving these bus hang-up problems.

Bus Error

A bus error occurs when a START or STOP condition is present at an illegal position in the format frame. Examples of illegal positions are during the serial transfer of an address byte, a data or an acknowledge bit.

The SIO1 hardware only reacts to a bus error when it is involved in a serial transfer either as a master or an addressed slave. When a bus error is detected, SIO1 immediately switches to the not addressed slave mode, releases the SDA and SCL lines, sets the interrupt flag, and loads the status register with 00H. This status code may be used to vector to a service routine which either attempts the aborted serial transfer again or simply recovers from the error condition as shown in Table 7.

Table 7. Miscellaneous States

STATUS CODE (S1STA)	STATUS OF THE I²C BUS AND SIO1 HARDWARE	APPLICATION SOFTWARE RESPONSE					NEXT ACTION TAKEN BY SIO1 HARDWARE
		TO/FROM S1DAT	TO S1CON				
			STA	STO	SI	AA	
F8H	No relevant state information available; SI = 0	No S1DAT action	No S1CON action				Wait or proceed current transfer
00H	Bus error during MST or selected slave modes, due to an illegal START or STOP condition. State 00H can also occur when interference causes SIO1 to enter an undefined state.	No S1DAT action	0	1	0	X	Only the internal hardware is affected in the MST or addressed SLV modes. In all cases, the bus is released and SIO1 is switched to the not addressed SLV mode. STO is reset.

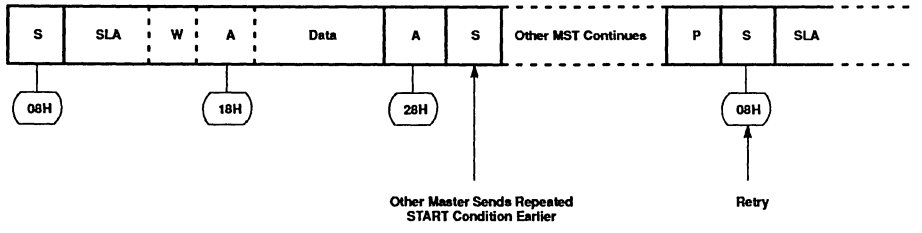


Figure 21. Simultaneous Repeated START Conditions from 2 Masters

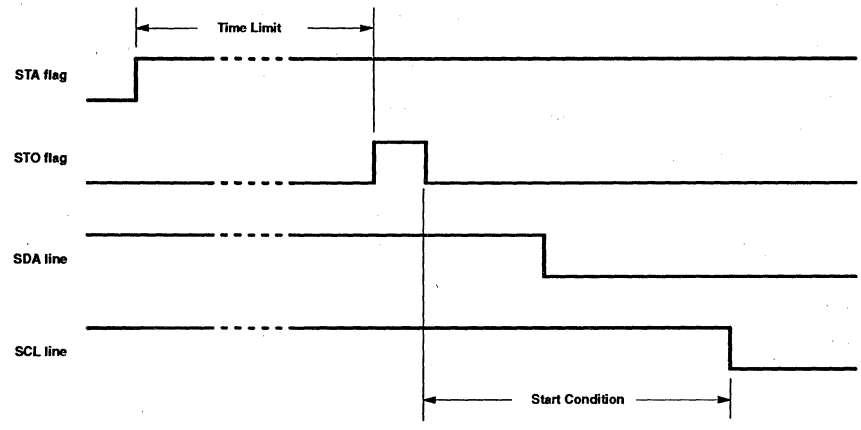


Figure 22. Forced Access to a Busy I²C Bus

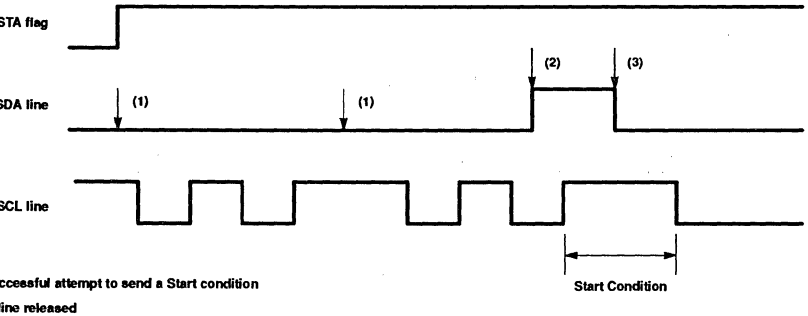


Figure 23. Recovering from a Bus Obstruction Caused by a Low Level on SDA

Software Examples of SIO1 Service Routines: This section consists of a software example for:

- Initialization of SIO1 after a RESET
- Entering the SIO1 interrupt routine
- The 26 state service routines for the
  - Master transmitter mode
  - Master receiver mode
  - Slave receiver mode
  - Slave transmitter mode

Initialization

In the initialization routine, SIO1 is enabled for both master and slave modes. For each mode, a number of bytes of internal data RAM are allocated to the SIO to act as either a transmission or reception buffer. In this example, 8 bytes of internal data RAM are reserved for different purposes. The data memory map is shown in Figure 24. The initialization routine performs the following functions:

- S1ADR is loaded with the part's own slave address and the general call bit (GC)
- P1.6 and P1.7 bit latches are loaded with logic 1s
- RAM location HADD is loaded with the high-order address byte of the service routines
- The SIO1 interrupt enable and interrupt priority bits are set
- The slave mode is enabled by simultaneously setting the ENS1 and AA bits in S1CON and the serial clock frequency (for master modes) is defined by loading CR0 and CR1 in S1CON. The master routines must be started in the main program.

The SIO1 hardware now begins checking the I²C bus for its own slave address and general call. If the general call or the own slave address is detected, an interrupt is requested and S1STA is loaded with the appropriate state information. The following text describes a fast method of branching to the appropriate service routine.

SIO1 Interrupt Routine

When the SIO1 interrupt is entered, the PSW is first pushed on the stack. Then S1STA and HADD (loaded with the high-order address byte of the 26 service routines by the initialization routine) are pushed on to the stack. S1STA contains a status code which is the lower byte of one of the 26 service routines. The next instruction is RET, which is the return from subroutine instruction. When this instruction is executed, the high and low order address bytes are popped from stack and loaded into the program counter.

The next instruction to be executed is the first instruction of the state service routine. Seven bytes of program code (which execute in eight machine cycles) are required to branch to one of the 26 state service routines.

```
SI      PUSH  PSW     Save PSW
        PUSH  S1STA   Push status code
                      (low order
                      address byte)
        PUSH  HADD    Push high order
                      address byte
        RET           Jump to state
                      service routine
```

The state service routines are located in a 256-byte page of program memory. The location of this page is defined in the initialization routine. The page can be located anywhere in program memory by loading data RAM register HADD with the page number. Page 01 is chosen in this example, and the service routines are located between addresses 0100H and 01FFH.

The State Service Routines

The state service routines are located 8 bytes from each other. Eight bytes of code are sufficient for most of the service routines. A few of the routines require more than 8 bytes and have to jump to other locations to obtain more bytes of code. Each state routine is part of the SIO1 interrupt routine and handles one of the 26 states. It ends with a RETI instruction which causes a return to the main program.

Master Transmitter and Master Receiver Modes

The master mode is entered in the main program. To enter the master transmitter mode, the main program must first load the internal data RAM with the slave address, data bytes, and the number of data bytes to be transmitted. To enter the master receiver mode, the main program must first load the internal data RAM with the slave address and the number of data bytes to be received. The R/$\overline{W}$ bit determines whether SIO1 operates in the master transmitter or master receiver mode.

Master mode operation commences when the STA bit in S1CION is set by the SETB instruction and data transfer is controlled by the master state service routines in accordance with Table 3, Table 4, Figure 17, and Figure 18. In the example below, 4 bytes are transferred. There is no repeated START condition. In the event of lost arbitration, the transfer is restarted when the bus becomes free. If a bus error occurs, the I^2C bus is released and SIO1 enters the not selected slave receiver mode. If a slave device returns a not acknowledge, a STOP condition is generated.

A repeated START condition can be included in the serial transfer if the STA flag is set instead of the STO flag in the state service routines vectored to by status codes 28H and 58H. Additional software must be written to determine which data is transferred after a repeated START condition.

Slave Transmitter and Slave Receiver Modes

After initialization, SIO1 continually tests the I^2C bus and branches to one of the slave state service routines if it detects its own slave address or the general call address (see Table 5, Table 6, Figure 19, and Figure 20). If arbitration was lost while in the master mode, the master mode is restarted after the current transfer. If a bus error occurs, the I^2C bus is released and SIO1 enters the not selected slave receiver mode.

In the slave receiver mode, a maximum of 8 received data bytes can be stored in the internal data RAM. A maximum of 8 bytes ensures that other RAM locations are not overwritten if a master sends more bytes. If more than 8 bytes are transmitted, a not acknowledge is returned, and SIO1 enters the not addressed slave receiver mode. A maximum of one received data byte can be stored in the internal data RAM after a general call address is detected. If more than one byte is transmitted, a not acknowledge is returned and SIO1 enters the not addressed slave receiver mode.

In the slave transmitter mode, data to be transmitted is obtained from the same locations in the internal data RAM that were previously loaded by the main program. After a not acknowledge has been returned by a master receiver device, SIO1 enters the not addressed slave mode.

Adapting the Software for Different Applications

The following software example shows the typical structure of the interrupt routine including the 26 state service routines and may be used as a base for user applications. If one or more of the four modes are not used, the associated state service routines may be removed but, care should be taken that a deleted routine can never be invoked.

This example does not include any time-out routines. In the slave modes, time-out routines are not very useful since, in these modes, SIO1 behaves essentially as a passive device. In the master modes, an internal timer may be used to cause a time-out if a serial transfer is not complete after a defined period of time. This time period is defined by the system connected to the I^2C bus.

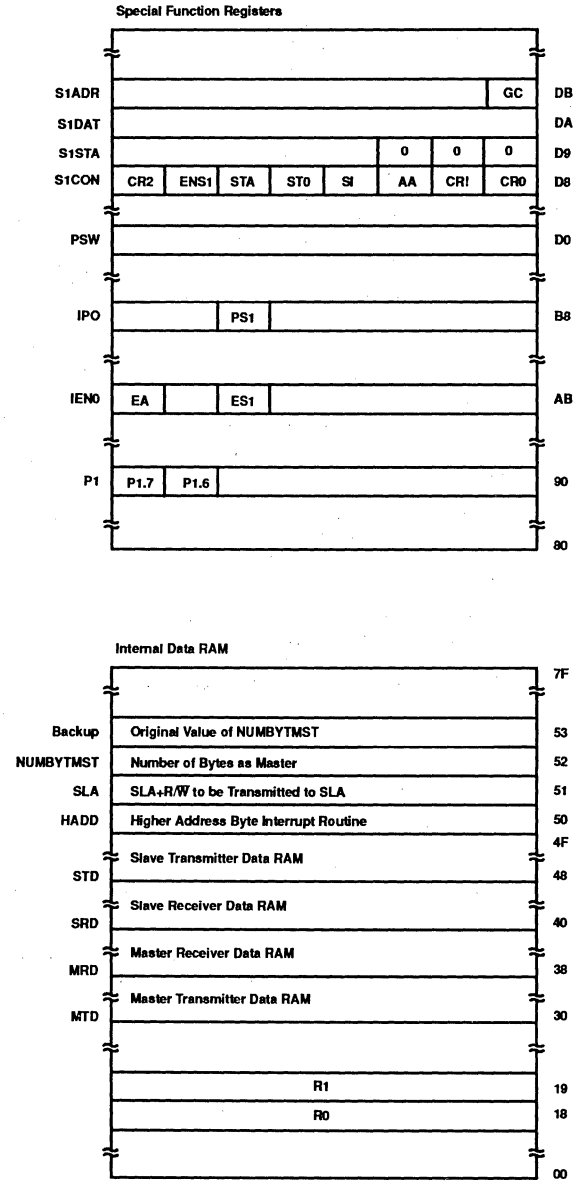


Figure 24. SIO1 Data Memory Map

```
                        !*****************************************************************************
                        ! SI01 EQUATE LIST
                        !*****************************************************************************
                        !*****************************************************************************
                        ! LOCATIONS OF THE SI01 SPECIAL FUNCTION REGISTERS
                        !*****************************************************************************
00D8                    S1CON   –0xd8
00D9                    S1STA   –0xd9
00DA                    S1DAT   –0xda
00DB                    S1ADR   –0xdb

00A8                    IEN0    –0xa8
00B8                    IP0     –02b8

                        !*****************************************************************************
                        ! BIT LOCATIONS
                        !*****************************************************************************

00DD                    STA     –0xdd           ! STA bit in S1CON
00BD                    SI01HP  –0xbd           ! IP0, SI01 Priority bit

                        !*****************************************************************************
                        ! IMMEDIATE DATA TO WRITE INTO REGISTER S1CON
                        !*****************************************************************************
00D5                    ENS1_NOTSTA_STO_NOTSI_AA_CR0            –0xd5   ! Generates STOP
                                                                        ! (CR0 = 100kHz)
00C5                    ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0         –0xc5   ! Releases BUS and
                                                                        ! ACK
00C1                    ENS1_NOTSTA_NOTSTO_NOTSI_NOTAA_CR0      –0xc1   ! Releases BUS and
                                                                        ! NOT ACK
00E5                    ENS1_STA_NOTSTO_NOTSI_AA_CR0            –0xe5   ! Releases BUS and
                                                                        ! set STA

                        !*****************************************************************************
                        ! GENERAL IMMEDIATE DATA
                        !*****************************************************************************
0031                    OWNSLA  –0x31           ! Own SLA+General Call
                                                ! must be written into S1ADR
00A0                    ENSI01  –0xa0           ! EA+ES1, enable SIO1 interrupt
                                                ! must be written into IEN0
0001                    PAG1    –0x01           ! select PAG1 as HADD
00C0                    SLAW    –0xc0           ! SLA+W to be transmitted
00C1                    SLAR    –0xc1           ! SLA+R to be transmitted
0018                    SELRB3  –0x18           ! Select Register Bank 3

                        !*****************************************************************************
                        ! LOCATIONS IN DATA RAM
                        !*****************************************************************************
0030                    MTD     –0x30           ! MST/TRX/DATA base address
0038                    MRD     –0x38           ! MST/REC/DATA base address
0040                    SRD     –0x40           ! SLV/REC/DATA base address
0048                    STD     –0x48           ! SLV/TRX/DATA base address

0053                    BACKUP      –0x53       ! Backup from NUMBYTMST
                                                ! To restore NUMBYTMST in case
                                                ! of an Arbitration Loss.
0052                    NUMBYTMST –0x52         ! Number of bytes to transmit
                                                ! or receive as MST.
0051                    SLA         –0x51       ! Contains SLA+R/W to be
                                                ! transmitted.
0050                    HADD        –0x50       ! High Address byte for STATE 0
                                                ! till STATE 25.
```

```
                                  !*****************************************************************************
                                  ! INITIALIZATION ROUTINE
                                  ! Example to initialize IIC Interface as slave receiver or
                                  ! slave transmitter and start a MASTER TRANSMIT or a MASTER
                                  ! RECEIVE function. 4 bytes will be transmitted or received.
                                  !*****************************************************************************
                                  .sect  strt
                                  .base  0x00
0000    4100                                  ajmp  INIT                  ! RESET

                                  .sect  initial
                                  .base  0x200
0200    75DB31                    INIT:       mov   S1ADR,#OWNSLA         ! Load own SLA + enable
                                                                          ! general call recognition
0203    D296                                  setb  P1(6)                 ! P1.6 High level.
0205    D297                                  setb  P1(7)                 ! P1.7 High level.
0207    755001                                mov   HADD,#PAG1
020A    43A8A0                                orl   IEN0,#ENSI01          ! Enable SI01 interrupt
020D    C2BD                                  clr   SI01HP                ! SI01 interrupt low
                                                                          ! priority
020F    75D8C5                                mov   S1CON, #ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                          ! Initialize SLV funct.

                                  !*****************************************************************************

                                  !-----------------------------------------------------------------
                                  ! START MASTER TRANSMIT FUNCTION
                                  !-----------------------------------------------------------------

0212    755204                                mov   NUMBYTMST,#0x4        ! Transmit 4 bytes.
0215    7551C0                                mov   SLA,#SLAW             ! SLA+W, Transmit funct.
0218    D2DD                                  setb  STA                   ! set STA in S1CON

                                  !-----------------------------------------------------------------
                                  ! START MASTER RECEIVE FUNCTION
                                  !-----------------------------------------------------------------
021A    755204                                mov   NUMBYTMST,#0x4        ! Receive 4 bytes.
021D    7551C1                                mov   SLA,#SLAR             ! SLA+R, Receive funct.
0220    D2DD                                  setb  STA                   ! set STA in S1CON
```

```
                        !*****************************************************************************
                        ! SI01 INTERRUPT ROUTINE
                        !*****************************************************************************
                        .sect  intvec                                   ! SI01 interrupt vector
                        .base  0x00

                        ! S1STA and HADD are pushed onto the stack. They serve as
                        ! return address for the RET instruction.
                        ! The RET instruction sets the Program Counter to address
                        ! HADD, S1STA and jumps to the right subroutine.

002B    C0D0                        push   psw                          ! save psw
002D    C0D9                        push   S1STA
002F    C050                        push   HADD
0031    22                          ret                                 ! JMP to address
                                                                        ! HADD,S1STA.

                        !-----------------------------------------------------------------------------
                        ! STATE  :  00, Bus error.
                        ! ACTION:  Enter not addressed SLV mode and release bus.
                                    STO reset.
                        !-----------------------------------------------------------------------------
                        .sect  st0
                        .base  0x100

0100    75D8D5                      mov    S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0  ! clr SI
                                                                                ! set STO,AA
0103    D0D0                        pop    psw
0105    32                          reti
```

```
                        !*****************************************************************************
                        !*****************************************************************************
                        ! MASTER STATE SERVICE ROUTINES
                        !*****************************************************************************
                        ! State 08 and State 10 are both for MST/TRX and MST/REC.
                        ! The R/W bit decides whether the next state is within
                        ! MST/TRX mode or within MST/REC mode.
                        !*****************************************************************************

                        !-----------------------------------------------------------
                        ! STATE  :  08, A, START condition has been transmitted.
                        ! ACTION:   SLA+R/W are transmitted, ACK bit is received.
                        !-----------------------------------------------------------
                        .sect  mts8
                        .base  0x108

0108   8551DA                           mov   S1DAT,SLA            ! Load SLA+R/W
010B   75D8C5                           mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                   ! clr SI
010E   01A0                             ajmp  INITBASE1

                        !-----------------------------------------------------------
                        ! STATE  :  10, A repeated START condition has been
                        !           transmitted.
                        ! ACTION:   SLA+R/W are transmitted, ACK bit is received.
                        !-----------------------------------------------------------
                        .sect  mts10
                        .base  0x110

0110   8551DA                           mov   S1DAT,SLA            ! Load SLA+R/W
0113   75D8C5                           mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                   ! clr SI
010E   01A0                             ajmp  INITBASE1

                        .sect  ibase1
                        .base  0xa0
00A0   75D018           INITBASE1:      mov   psw,#SELRB3
00A3   7930                             mov   r1,#MTD
00A5   7838                             mov   r0,#MRD
00A7   855253                           mov   BACKUP,NUMBYTMST          ! Save initial value
00AA   D0D0                             pop   psw
00AC   32                               reti
```

```
                        !*****************************************************************************
                        !*****************************************************************************
                        ! MASTER TRANSMITTER STATE SERVICE ROUTINES
                        !*****************************************************************************
                        !*****************************************************************************

                        !-----------------------------------------------------
                        ! STATE  :  18, Previous state was STATE 8 or STATE 10,
                        !           SLA+W have been transmitted, ACK has
                        !           been received.
                        ! ACTION:   First DATA is transmitted, ACK bit
                        !           is received.
                        !-----------------------------------------------------
                        .sect  mts18
                        .base  0x118

0118  75D018                        mov   psw,#SELRB3
011B  87DA                          mov   S1DAT,@r1
011D  01B5                          ajmp  CON

                        !-----------------------------------------------------
                        ! STATE  :  20, SLA+W have been transmitted, NOT ACK
                        !           has been received
                        ! ACTION:   Transmit STOP condition.
                        !-----------------------------------------------------
                        .sect  mts20
                        .base  0x110

0120  75D8D5                        mov   S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0
                                                          ! set STO, clr SI
0123  D0D0                          pop   psw
0125  32                            reti

                        !-----------------------------------------------------
                        ! STATE  :  28, DATA of S1DAT have been transmitted,
                        !           ACK received.
                        ! ACTION:   If Transmitted DATA is last DATA then
                        !           transmit a STOP condition, else transmit
                        !           next DATA.
                        !-----------------------------------------------------
                        .sect  mts28
                        .base  0x128

0128  D55285                        djnz  NUMBYTMST,NOTLDAT1      ! JMP if NOT last DATA
012B  75D8D5                        mov   S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0
                                                          ! clr SI, set AA
012E  01B9                          ajmp  RETmt

                        .sect  mts28sb
                        .base  0x0b0
00B0  75D018            NOTLDAT1:   mov   psw,#SELRB3
00B3  87DA                          mov   S1DAT,@r1
00B5  75D8C5            CON:        mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                          ! clr SI, set AA
00B8  09                            inc   r1
00B9  D0D0              RETmt    :  pop   psw
00BB  32                            reti
```

```
                        ;-------------------------------------------------
                        ; STATE  :  30, DATA of S1DAT have been transmitted,
                        ;          NOT ACK received.
                        ; ACTION:  Transmit a STOP condition.
                        ;-------------------------------------------------
                        .sect  mts30
                        .base  0x130

0130  75D8D5                      mov   S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0
                                                        ; set STO, clr SI
0133  D0D0                        pop   psw
0135  32                          reti

                        ;-------------------------------------------------
                        ; STATE  :  38, Arbitration lost in SLA+W or DATA.
                        ; ACTION:  Bus is released, not addressed SLV mode is
                        ;          entered. A new START condition is transmitted
                        ;          when the IIC bus is free again.
                        ;-------------------------------------------------
                        .sect  mts38
                        .base  0x138

0138  75D8E5                      mov   S1CON,#ENS1_STA_NOTSTO_NOTSI_AA_CR0
013B  855352                      mov   NUMBYTMST,BACKUP
013E  01B9                        ajmp  RETmt


                        ;*************************************************
                        ;*************************************************
                        ; MASTER RECEIVER STATE SERVICE ROUTINES
                        ;*************************************************
                        ;*************************************************


                        ;-------------------------------------------------
                        ; STATE  :  40, Previous state was STATE 08 or STATE 10,
                        ;          SLA+R have been transmitted, ACK received.
                        ; ACTION:  DATA will be received, ACK returned.
                        ;-------------------------------------------------
                        .sect  mts40
                        .base  0x140

0140  75D8C5                      mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                        ; clr STA, STO, SI set AA
0143  01CB                        ajmp  RETmr

                        ;-------------------------------------------------
                        ; STATE  :  48, SLA+R have been transmitted, NOT ACK
                        ;          received.
                        ; ACTION:  STOP condition will be generated.
                        ;-------------------------------------------------
                        .sect  mts48
                        .base  0x148

0148  75D8D5                      mov   S1CON,#ENS1_NOTSTA_STO_NOTSI_AA_CR0
                                                        ; set STO, clr SI
014B  D0D0                        pop   psw
014D  32                          reti
```

```
                        !-----------------------------------------------------
                        ! STATE  :  50, DATA have been received, ACK returned.
                        ! ACTION:  Read DATA of S1DAT.
                        !          DATA will be received, if it is last
                        !          DATA then NOT ACK will be returned else ACK
                        !          will be returned.
                        !-----------------------------------------------------
                        .sect  mrs50
                        .base  0x150

0150  75D018                        mov   psw,#SELRB3
0153  A6DA                          mov   @r0,S1DAT          ! Read received DATA
0155  01C0                          ajmp  REC1

                        .sect  mrs50s
                        .base  0xc0

00C0  D55205            REC1:       djnz  NUMBYTMST,NOTLDAT2
00C3  75D8C1                        mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_NOTAA_CR0
                                                             ! clr SI,AA
00C6  8003                          sjmp  RETmr
00C8  75D8C5            NOTLDAT2:   mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                             ! clr SI, set AA
00CB  08                RETmr:      inc   r0
00CC  D0D0                          pop   psw
00CE  32                            reti

                        !-----------------------------------------------------
                        ! STATE  :  58, DATA have been received, NOT ACK
                        !          returned.
                        ! ACTION:  Read DATA of S1DAT and generate a STOP
                        !          condition.
                        !-----------------------------------------------------
                        .sect  mrs58
                        .base  0x158

0158  75D018                        mov   psw,#SELRB3
015B  A6DA                          mov   @0,S1DAT
015D  80E9                          sjmp  STOP
```

```
                                        !*****************************************************************************
                                        !*****************************************************************************
                                        ! SLAVE RECEIVER STATE SERVICE ROUTINES
                                        !*****************************************************************************
                                        !*****************************************************************************

                                        !-----------------------------------------------------------------------------
                                        ! STATE  :  60, Own SLA+W have been received, ACK
                                        !           returned.
                                        ! ACTION:   DATA will be received and ACK returned.
                                        !-----------------------------------------------------------------------------
                                        .sect  srs60
                                        .base  0x160
0160    75D8C5                                  mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                        ! clr  SI, set  AA
0163    75D018                                  mov   psw,#SELRB3
0166    01D0                                    ajmp  INITSRD
                                        .sect  insrd
                                        .base  0xd0

00D0    7840                            INITSRD:        mov   r0,#SRD
00D2    7908                                    mov   r1,#8
00D4    D0D0                                    pop   psw
00D6    32                                      reti

                                        !-----------------------------------------------------------------------------
                                        ! STATE  :  68, Arbitration lost in SLA and R/W as MST
                                        !           Own SLA+W have been received, ACK returned
                                        ! ACTION:   DATA will be received and ACK returned.
                                        !           STA is set to restart MST mode after the
                                        !           bus is free again.
                                        !-----------------------------------------------------------------------------
                                        .sect  srs68
                                        .base  0x168
0168    75D8E5                                  mov   S1CON,#ENS1_STA_NOTSTO_NOTSI_AA_CR0
016B    75D018                                  mov   psw,#SELRB3
016E    01D0                                    ajmp  INITSRD

                                        !-----------------------------------------------------------------------------
                                        ! STATE  :  70, General call has been received, ACK
                                        !           returned.
                                        ! ACTION:   DATA will be received and ACK returned.
                                        !-----------------------------------------------------------------------------
                                        .sect  srs70
                                        .base  0x170
0170    75D8C5                                  mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                        ! clr  SI, set  AA
0173    75D018                                  mov   psw,#SELRB3       ! Initialize SRD counter

                                        !-----------------------------------------------------------------------------
                                        ! STATE  :  78, Arbitration lost in SLA+R/W as MST.
                                        !           General call has been received, ACK returned.
                                        ! ACTION:   DATA will be received and ACK returned.
                                        !           STA is set to restart MST mode after the
                                        !           bus is free again.
                                        !-----------------------------------------------------------------------------
                                        .sect  srs78
                                        .base  0x178
0178    75D8E5                                  mov   S1CON,#ENS1_STA_NOTSTO_NOTSI_AA_CR0
017B    75D018                                  mov   psw,#SELRB3               ! Initialize SRD counter
017E    01D0                                    ajmp  INITSRD
```

```
                                        !-----------------------------------------------------
                                        ! STATE  :  80, Previously addressed with own SLA.
                                        !           DATA received, ACK returned.
                                        ! ACTION :  Read DATA.
                                        !           IF received DATA was the last THEN superfluous
                                        !           DATA will be received and NOT ACK returned ELSE
                                        !           next DATA will be received and ACK returned.
                                        !-----------------------------------------------------
                                        .sect  srs80
                                        .base  0x180

0180   75D018                                   mov   psw,#SELRB3
0183   A6DA                                     mov   @r0,S1DAT          ! Read received DATA
0185   01D8                                     ajmp  REC2

                                        .sect  srs80s
                                        .base  0xd8

00D8   D906                             REC2:       djnz  r1,NOTLDAT3
00DA   75D8C1                                   mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_NOTAA_CR0
                                                                        ! clr  SI,AA
00DD   D0D0                                     pop   psw
00DF   32                                       reti
00E0   75D8C5                           NOTLDAT2:   mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                        ! clr  SI, set  AA
00E3   08                                       inc   r0
00E4   D0D0                             RETsr:      pop   psw
00E6   32                                       reti


                                        !-----------------------------------------------------
                                        ! STATE  :  88, Previously addressed with own SLA.
                                        !           DATA received NOT ACK returned.
                                        ! ACTION :  No save of DATA, Enter NOT addressed SLV
                                        !           mode. Recognition of own SLA. General call
                                        !           recognized, if S1ADR. 0–1.
                                        !-----------------------------------------------------
                                        .sect  srs88
                                        .base  0x188

0188   75D8C5                                   mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                                        ! clr  SI, set  AA
018B   01E4                                     ajmp  RETsr

                                        !-----------------------------------------------------
                                        ! STATE  :  90, Previously addressed with general call.
                                        !           DATA has been received, ACK has been returned.
                                        ! ACTION :  Read DATA.
                                        !           After General call only one byte will be
                                        !           received with ACK the second DATA will be
                                        !           received with NOT ACK.
                                        !           DATA will be received and NOT ACK
                                        !           returned.
                                        !-----------------------------------------------------
                                        .sect  srs90
                                        .base  0x190

0190   75D018                                   mov   psw,#SELRB3
0193   A6DA                                     mov   @r0,S1DAT          ! Read received DATA
0195   01DA                                     ajmp  LDAT
```

```
                        ;-------------------------------------------------------
                        ! STATE  :  98, Previously addressed with general call.
                        !          DATA has been received, NOT ACK has been
                        !          returned.
                        ! ACTION:   No save of DATA, Enter NOT addressed SLV
                        !          mode. Recognition of own SLA. General call
                        !          recognized, if S1ADR. 0–1.
                        ;-------------------------------------------------------
                        .sect  srs98
                        .base  0x198

0198  75D8C5                      mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                        ! clr SI, set AA
019B  D0D0                        pop   psw
019D  32                          reti

                        ;-------------------------------------------------------
                        ! STATE  :  A0, A STOP condition or repeated START has
                        !          been received, while still addressed as
                        !          SLV/REC or SLV/TRX.
                        ! ACTION:   No save of DATA, Enter NOT addressed SLV
                        !          mode. Recognition of own SLA. General call
                        !          recognized, if S1ADR. 0–1.
                        ;-------------------------------------------------------
                        .sect  srsA0
                        .base  0x1a0

01A0  75D8C5                      mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                        ! clr SI, set AA
01A3  D0D0                        pop   psw
01A5  32                          reti
```

```
                        |**********************************************************************
                        |**********************************************************************
                        ! SLAVE TRANSMITTER STATE SERVICE ROUTINES
                        |**********************************************************************
                        |**********************************************************************

                        |-----------------------------------------------
                        ! STATE  :  A8, Own SLA+R received, ACK returned.
                        ! ACTION:  DATA will be transmitted, A bit received.
                        |-----------------------------------------------
                        .sect  stsa8
                        .base  0x1a8

01A8  8548DA                        mov   S1DAT,STD          ! load DATA in S1DAT
01AB  75D8C5                        mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                             ! clr SI, set AA
01AE  01E8                          ajmp  INITBASE2

                        .sect  ibase2
                        .base  0xe8
00E8  75D018            INITBASE2:  mov   psw,#SELRB3
00EB  7948                          mov   r1, #STD
00ED  09                            inc   r1
00EE  D0D0                          pop   psw
00F0  32                            reti

                        |-----------------------------------------------
                        ! STATE  :  B0, Arbitration lost in SLA and R/W as MST.
                        !           Own SLA+R received, ACK returned.
                        ! ACTION:  DATA will be transmitted, A bit received.
                        !           STA is set to restart MST mode after the
                        !           bus is free again.
                        |-----------------------------------------------
                        .sect  stsb0
                        .base  0x1b0

01B0  8548DA                        mov   S1DAT,STD          ! load DATA in S1DAT
01B3  75D8E5                        mov   S1CON,#ENS1_STA_NOTSTO_NOTSI_AA_CR0
01B6  01E8                          ajmp  INITBASE2

                        |-----------------------------------------------
                        ! STATE  :  B8, DATA has been transmitted, ACK received.
                        ! ACTION:  DATA will be transmitted, ACK bit is received.
                        |-----------------------------------------------
                        .sect  stsb8
                        .base  0x1b8
01B8  75D018                        mov   psw,#SELRB3
01BB  87DA                          mov   S1DAT,@r1
01BD  01F8                          ajmp  SCON

                        .sect  scn
                        .base  0xf8

00F8  75D8C5            SCON:       mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                             ! clr SI, set AA

00FB  09                            inc   r1
00FC  D0D0                          pop   psw
00FE  32                            reti
```

```
                        |-------------------------------------------------
                        ! STATE  :  C0, DATA has been transmitted, NOT ACK
                        !           received.
                        ! ACTION:   Enter not addressed SLV mode.
                        |-------------------------------------------------
                        .sect  stsc0
                        .base  0x1c0
01C0   75D8C5                       mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                          !clr SI, set AA
01C3   D0D0                         pop   psw
01C5   32                           reti

                        |-------------------------------------------------
                        ! STATE  :  C8, Last DATA has been transmitted (AA=0),
                        !           ACK received.
                        ! ACTION:   Enter not addressed SLV mode.
                        |-------------------------------------------------
                        .sect  stsc8
                        .base  0x1c8
01C8   75D8C5                       mov   S1CON,#ENS1_NOTSTA_NOTSTO_NOTSI_AA_CR0
                                                          !clr SI, set AA
01CB   D0D0                         pop   psw
01CD   32                           reti

                        |*************************************************************************************
                        |*************************************************************************************
                        ! END OF SI01 INTERRUPT ROUTINE
                        |*************************************************************************************
                        |*************************************************************************************
```

Reset Circuitry

The reset circuitry for the 8XC552 is connected to the reset pin RST. A Schmitt trigger is used at the input for noise rejection (see Figure 25). The output of the Schmitt trigger is sampled by the reset circuitry every machine cycle.

A reset is accomplished by holding the RST pin HIGH for at least two machine cycles (24 oscillator periods) while the oscillator is running. The CPU responds by executing an internal reset. During reset, ALE and PSEN output a HIGH level. In order to perform a correct reset, this level must not be affected by external elements. The RST line can also be pulled HIGH internally by a pull-up transistor activated by the watchdog timer T3. The length of the output pulse from T3 is 3 machine cycles. A pulse of such short duration is necessary in order to recover from a processor or system fault as fast as possible.

Note that the short reset pulse from Timer T3 cannot discharge the power-on reset capacitor (see Figure 26). Consequently, when the watchdog timer is also used to set external devices, this capacitor arrangement should not be connected to the RST pin, and a different circuit should be used to perform the power-on reset operation. A timer T3 overflow, if enabled, will force a reset condition to the 8XC552 by an internal connection, whether the output RTS is tied LOW or not.

The internal reset is executed during the second cycle in which RST is HIGH and is repeated every cycle until RST goes low. It leaves the internal registers as follows:

REGISTER	CONTENT	
ACC	0000	0000
ADCON	xx00	0000
ADCH	xxxx	xxxx
B	0000	0000
CML0-CML2	0000	0000
CMH0-CMH2	0000	0000
CTCON	0000	0000
CTL0-CTL3	xxxx	xxxx
CTH0-CTH3	xxxx	xxxx
DPL	0000	0000
DPH	0000	0000
IEN0	0000	0000
IEN1	0000	0000
IP0	x000	0000
IP1	0000	0000
PCH	0000	0000
PCL	0000	0000
PCON	0xx0	0000
PSW	0000	0000
PWM0	0000	0000
PWM1	0000	0000
PWMP	0000	0000
P0-P4	1111	1111
P5	xxxx	xxxx
RTE	0000	0000
S0BUF	xxxx	xxxx
S0CON	0000	0000
S1ADR	0000	0000
S1CON	0000	0000
S1DAT	0000	0000
S1STA	1111	1000
SP	0000	0111
STE	1100	0000
TCON	0000	0000
TH0, TH1	0000	0000
TMH2	0000	0000
TL0, TL1	0000	0000
TML2	0000	0000
TMOD	0000	0000
TM2CON	0000	0000
TM2IR	0000	0000
T3	0000	0000

The internal RAM is not affected by reset. At power-on, the RAM content is indeterminate.

Interrupts

The 8XC552 has fifteen interrupt sources, each of which can be assigned one of two priority levels, as shown in Figure 27. The five interrupt sources common to the 80C51 are the external interrupts ($\overline{INT0}$ and $\overline{INT1}$), the timer 0 and timer 1 interrupts (IT0 and IT1), and the serial I/O interrupt (RI or TI). In the 8XC552, the standard serial interrupt is called SIO0. Since the subsystems which create these interrupts are identical on both parts, their functionality is likewise identical. The only differences are the locations of the enable and priority register configurations and the priority structure. This is detailed below along with the specifics of the interrupts unique to the 8XC552.

The eight Timer T2 interrupts are generated by flags CTI0-CT13, CMI0-CMI2, and by the logical OR of flags T2OV and T2BO. Flags CTI0 to CT13 are set by input signals CT0I to CT3i. Flags CMI0 to CMI2 are set when a match occurs between Timer T2 and the compare registers CM0, CM1, and CM2. When an 8-bit or 16-bit overflow occurs, flags T2BO and T2OV are set, respectively. These nine flags are not cleared by hardware and must be reset by software to avoid recurring interrupts.

The ADC interrupt is generated by the ADCI flag in the ADC control register (ADCON). This flag is set when an ADC conversion result is ready to be read. ADCI is not cleared by hardware and must be reset by software to avoid recurring interrupts.

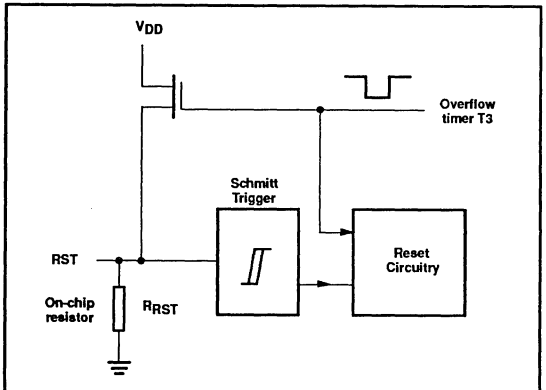


Figure 25. On-Chip Reset Configuration

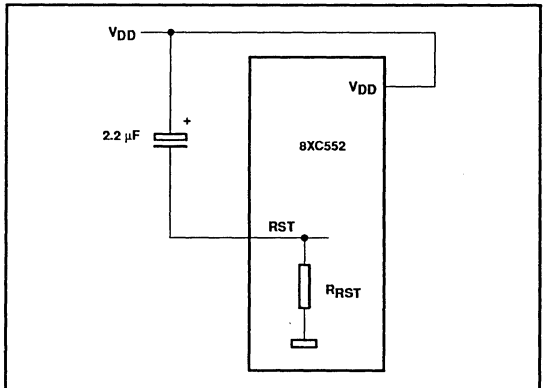


Figure 26. Power-On Reset

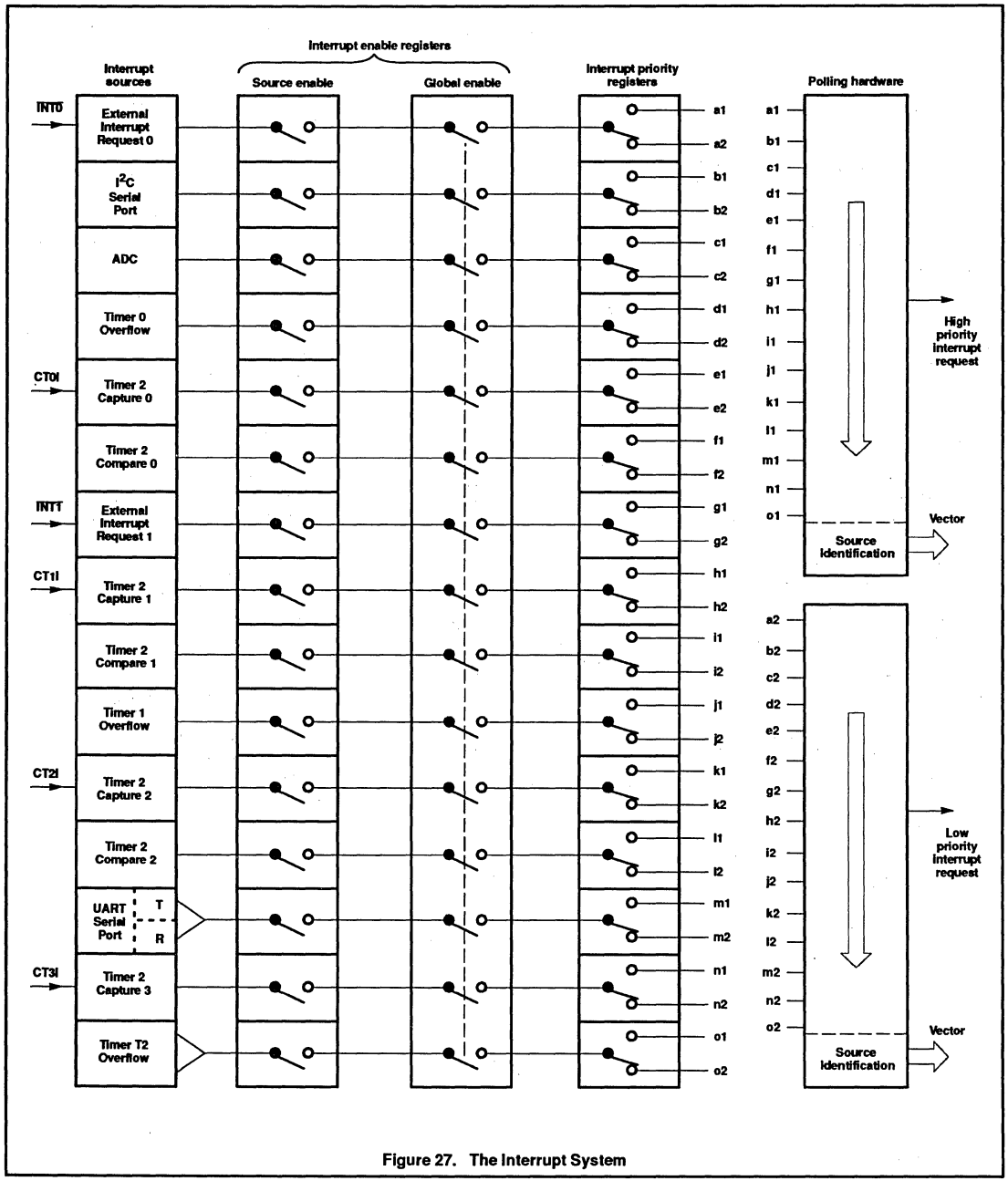


Figure 27. The Interrupt System

The SIO1 (I^2C) interrupt is generated by the SI flag in the SIO1 control register (S1CON). This flag is set when S1STA is loaded with a valid status code.

The ADCI flag may be reset by software. It cannot be set by software. All other flags that generate interrupts may be set or cleared by software, and the effect is the same as setting or resetting the flags by hardware. Thus, interrupts may be generated by software and pending interrupts can be canceled by software.

Interrupt Enable Registers: Each interrupt source can be individually enabled or disabled by setting or clearing a bit in the interrupt enable special function registers IEN0 and IEN1. All interrupt sources can also be globally enabled or disabled by setting or clearing bit EA in IEN0. The interrupt enable registers are described in Figures 28 and 29.

Interrupt Priority Structure: Each interrupt source can be assigned one of two priority levels. Interrupt priority levels are defined by the interrupt priority special function registers IP0 and IP1. IP0 and IP1 are described in Figures 30 and 31.

Interrupt priority levels are as follows:
"0"—low priority
"1"—high priority

A low priority interrupt may be interrupted by a high priority interrupt. A high priority interrupt cannot be interrupted by any other interrupt source. If two requests of different priority occur simultaneously, the high priority level request is serviced. If requests of the same priority are received simultaneously, an internal polling sequence determines which request is serviced. Thus, within each priority level, there is a second priority structure determined by the polling sequence. This second priority structure is shown in Table 8.

The above Priority Within Level structure is only used when there are simultaneous requests of the same priority level.

Interrupt Handling: The interrupt sources are sampled at S5P2 of every machine cycle. The samples are polled during the following machine cycle. If one of the flags was in a set condition at S5P2 of the previous machine cycle, the polling cycle will find it and the interrupt system will generate an LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1. An interrupt of higher or equal priority level is already in progress.
2. The current machine cycle is not the final cycle in the execution of the instruction in progress. (No interrupt request will be serviced until the instruction in progress is completed.)
3. The instruction in progress is RETI or any access to the interrupt priority or interrupt enable registers. (No interrupt will be serviced after RETI or after a read or write to IP0, IP1, IE0, or IE1 until at least one other instruction has been subsequently executed.)

The polling cycle is repeated with every machine cycle, and the values polled are the values present at S5P2 of the previous machine cycle. Note that if an interrupt flag is active but is not being responded to because of one of the above conditions, and if the flag is inactive when the blocking condition is removed, then the blocked interrupt will not be serviced. Thus, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

The processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate service routine. In some cases it also clears the flag which generated the interrupt, and in others it does not. It clears the Timer 0, Timer 1, and external interrupt flags. An external interrupt flag (IE0 or IE1) is cleared only if it was transition-activated. All other interrupt flags are not cleared by hardware and must be cleared by the software. The LCALL pushes the contents of the program counter on to the stack (but it does not save the PSW) and reloads the PC with an address that depends on the source of the interrupt being vectored to as shown in Table 9.

Execution proceeds from the vector address until the RETI instruction is encountered. The RETI instruction clears the "priority level active" flip-flop that was set when this interrupt was acknowledged. It then pops the top two bytes from the stack and reloads the program counter. Execution of the interrupted program continues from where it was interrupted.

	7	6	5	4	3	2	1	0
IEN0 (A8H)	EA	EAD	ES1	ES0	ET1	EX1	ET0	EX0
	(MSB)							(LSB)

Bit	Symbol	Function
IEN0.7	EA	Global enable/disable control 0 = No interrupt is enabled 1 = Any individually enabled interrupt will be accepted
IEN0.6	EAD	Enable ADC interrupt
IEN0.5	ES1	Enable SIO1 (I^2C) interrupt
IEN0.4	ES0	Enable SIO0 (UART) interrupt
IEN0.3	ET1	Enable Timer 1 interrupt
IEN0.2	EX1	Enable External interrupt 1
IEN0.1	ET0	Enable Timer 0 interrupt
IEN0.0	EX0	Enable External interrupt 0

Figure 28. Interrupt Enable Register (IEN0)

	7	6	5	4	3	2	1	0
IEN1 (E8H)	ET2	ECM2	ECM1	ECM0	ECT3	ECT2	ECT1	ECT0
	(MSB)							(LSB)

Bit	Symbol	Function
IEN1.7	ET2	Enable T2 overflow interrupt(s)
IEN1.6	ECM2	Enable T2 comparator 2 interrupt
IEN1.5	ECM1	Enable T2 comparator 1 interrupt
IEN1.4	ECM0	Enable T2 comparator 0 interrupt
IEN1.3	ECT3	Enable T2 capture register 3 interrupt
IEN1.2	ECT2	Enable T2 capture register 2 interrupt
IEN1.1	ECT1	Enable T2 capture register 1 interrupt
IEN1.0	ECT0	Enable T2 capture register 0 interrupt

In all cases, if the enable bit is 0, then the interrupt is disabled, and if the enable bit is 1, then the interrupt is enabled.

Figure 29. Interrupt Enable Register (IEN1)

	7	6	5	4	3	2	1	0
IP0 (B8H)	–	PAD	PS1	PS0	PT1	PX1	PT0	PX0
	(MSB)							(LSB)

Bit	Symbol	Function
IP.7	–	Unused
IP.6	PAD	ADC interrupt priority level
IP.5	PS1	SIO1 (I^2C) interrupt priority level
IP.4	PS0	SIO0 (UART) interrupt priority level
IP.3	PT1	Timer 1 interrupt priority level
IP.2	PX1	External interrupt 1 priority level
IP.1	PT0	Timer 0 interrupt priority level
IP.0	PX0	External interrupt 0 priority level

Figure 30. Interrupt Priority Register (IP0)

	7	6	5	4	3	2	1	0
IP1 (F8H)	PT2	PCM2	PCM1	PCM0	PCT3	PCT2	PCT1	PCT0
	(MSB)							(LSB)

Bit	Symbol	Function
IP1.7	PT2	T2 overflow interrupt(s) priority level
IP1.6	PCM2	T2 comparator 2 interrupt priority level
IP1.5	PCM1	T2 comparator 1 interrupt priority level
IP1.4	PCM0	T2 comparator 0 interrupt priority level
IP1.3	PCT3	T2 capture register 3 interrupt priority level
IP1.2	PCT2	T2 capture register 2 interrupt priority level
IP1.1	PCT1	T2 capture register 1 interrupt priority level
IP1.0	PCT0	T2 capture register 0 interrupt priority level

Figure 31. Interrupt Priority Register (IP1)

Table 8. Interrupt Priority Structure

SOURCE	NAME	PRIORITY WITHIN LEVEL
		(highest) ↑
External interrupt 0	X0	
SIO1 (I^2C)	S1	
ADC completion	ADC	
Timer 0 overflow	T0	
T2 capture 0	CT0	
T2 compare 0	CM0	
External interrupt 1	X1	
T2 capture 1	CT1	
T2 compare 1	CM1	
Timer 1 overflow	T1	
T2 capture 2	CT2	
T2 compare 2	CM2	
SIO0 (UART)	S0	
T2 capture 3	CT3	
Timer T2 overflow	T2	↓ (lowest)

Table 9. Interrupt Vector Addresses

SOURCE	NAME	VECTOR ADDRESS
External interrupt 0	X0	0003H
Timer 0 overflow	T0	000BH
External interrupt 1	X1	0013H
Timer 1 overflow	T1	001BH
SIO0 (UART)	S0	0023H
SIO1 (I^2C)	S1	002BH
T2 capture 0	CT0	0033H
T2 capture 1	CT1	003BH
T2 capture 2	CT2	0043H
T2 capture 3	CT3	004BH
ADC completion	ADC	0053H
T2 compare 0	CM0	005BH
T2 compare 1	CM1	0063H
T2 compare 2	CM2	006BH
T2 overflow	T2	0073H

I/O Port Structure
The 8XC552 has six 8-bit ports. Each port consists of a latch (special function registers P0 to P5), an input buffer, and an output driver (port 0 to 4 only). Ports 0-3 are the same as in the 80C51, with the exception of the additional functions of port 1. The parallel I/O function of port 4 is equal to that of ports 1, 2, and 3. Port 5 may be used as an input port only.

Figure 32 shows the bit latch and I/O buffer functional diagrams of the unique 8XC552 ports. A bit latch corresponds to one bit in a port's SFR and is represented as a D type flip-flop. A "write to latch" signal from the CPU latches a bit from the internal bus and a "read latch" signal from the CPU places the Q output of the flip-flop on the internal bus. A "read pin" signal from the CPU places the actual port pin level on the internal bus. Some instructions that read a port read the actual port pin levels, and other instructions read the latch (SFR) contents.

Port 1 Operation
Port 1 operates the same as it does in the 8051 with the exception of port lines P1.6 and P1.7, which may be selected as the SCL and SDA lines of serial port SIO1 (I^2C). Because the I^2C bus may be active while the device is disconnected from V_{DD}, these pins are provided with open drain drivers. Therefore pins P1.6 and P1.7 do not have internal pull-ups.

Port 5 Operation
Port 5 may be used to input up to 8 analog signals to the ADC. Unused ADC inputs may be used to input digital inputs. These inputs have an inherent hysteresis to prevent the input logic from drawing excessive current from the power lines when driven by analog signals. Channel to channel crosstalk (Ct) should be taken into consideration when both analog and digital signals are simultaneously input to Port 5 (see, D.C. characteristics in data sheet).

Port 5 is not bidirectional and may not be configured as an output port. All six ports are multifunctional, and their alternate functions are listed in Table 10. A more detailed description of these features can be found in the relevant parts of this section.

Pulse Width Modulated Outputs
The 8XC552 contains two pulse width modulated output channels (see Figure 33). These channels generate pulses of programmable length and interval. The repetition frequency is defined by an 8-bit prescaler PWMP, which supplies the clock for the counter. The prescaler and counter are common to both PWM channels. The 8-bit counter counts modulo 255, i.e., from 0 to 254 inclusive. The value of the 8-bit counter is compared to the contents of two registers: PWM0 and PWM1. Provided the contents of either of these registers is greater than the counter value, the corresponding $\overline{PWM0}$ or $\overline{PWM1}$ output is set LOW. If the contents of these registers are equal to, or less than the counter value, the output will be HIGH. The pulse-width-ratio is therefore defined by the contents of the registers PWM0 and PWM1. The pulse-width-ratio is in the range of 0 to 1 and may be programmed in increments of 1/255.

Buffered PWM outputs may be used to drive DC motors. The rotation speed of the motor would be proportional to the contents of PWMn. The PWM outputs may also be configured as a dual DAC. In this application, the PWM outputs must be integrated using conventional operational amplifier circuitry. If the resulting output voltages have to be accurate, external buffers with their own analog supply should be used to buffer the PWM outputs before they are integrated. The repetition frequency fpwm, at the PWMn outputs is give by:

$$fpwm = \frac{fosc}{2 \times (1 + PWMP) \times 255}$$

This gives a repetition frequency range of 123Hz to 31.4kHz (fosc = 16MHz). At fosc = 24MHz, the frequency range is 184Hz to 47.1Hz. By loading the PWM registers with either 00H or FFH, the PWM channels will output a constant HIGH or LOW level, respectively. Since the 8-bit counter counts modulo 255, it can never actually reach the value of the PWM registers when they are loaded with FFH.

When a compare register (PWM0 or PWM1) is loaded with a new value, the associated output is updated immediately. It does not have to wait until the end of the current counter period. Both $\overline{PWMn}$ output pins are driven by push-pull drivers. These pins are not used for any other purpose.

Prescaler frequency control register PWMP

PWMP (FEH)	7	6	5	4	3	2	1	0
	MSB							LSB

PWMP.0-7 Prescaler division factor = PWMP + 1.

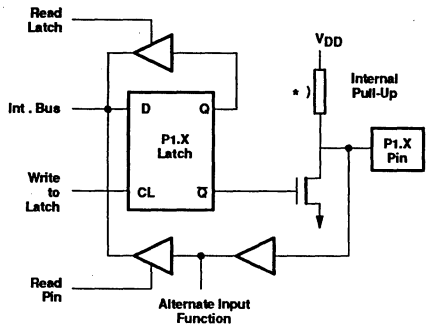
Reading PWMP gives the current reload value. The actual count of the prescaler cannot be read.

PWM0 (FCH) PWM1 (FDH)	7	6	5	4	3	2	1	0
	MSB							LSB

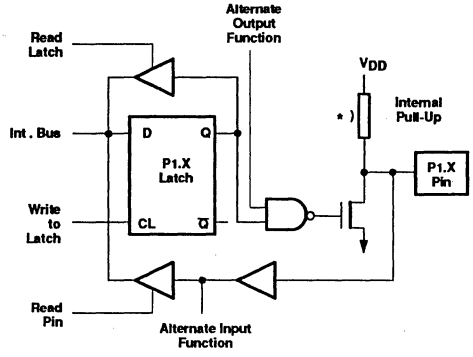
PWM0/1.0-7} Low/high ratio of $\overline{PWMn}$ =

$$\frac{(PWMn)}{255 - (PWMn)}$$

Analog-to-Digital Converter
The analog input circuitry consists of an 8-input analog multiplexer and a 10-bit, straight binary, successive approximation ADC. The analog reference voltage and analog power supplies are connected via separate input pins. The conversion takes 50 machine cycles, i.e., 37.5µs at an oscillator frequency of 16MHz, 25µs at an oscillator frequency of 24MHz. Input voltage swing is from 0V to +5V. Because the internal DAC employs a ratiometric potentiometer, there are no discontinuities in the converter characteristic. Figure 34 shows a functional diagram of the analog input circuitry.

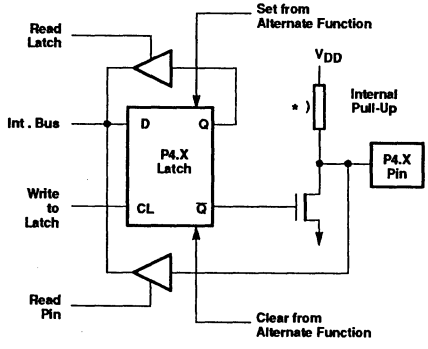


A.

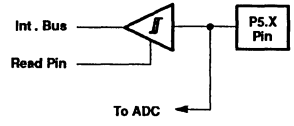


B.

NOTE:
Pull-up not present on P1.6 and P1.7.
*Two period active pull-up as in the 80C51.



C.



D.

Figure 32. Port Bit Latches and I/O Buffers

Table 10. Input/Output Ports

PORT PIN	ALTERNATE FUNCTION	
P0.0	AD0	Multiplexed lower order address/data bus used during external memory accesses
P0.1	AD1	
P0.2	AD2	
P0.3	AD3	
P0.4	AD4	
P0.5	AD5	
P0.6	AD6	
P0.7	AD7	
P1.0	CT0I	Capture timiner input signals for timer T2
P1.1	CT1I	
P1.2	CT2I	
P1.3	CT3I	
P1.4	T2	T2 event input
P1.5	RT2	T2 timer reset signal. Rising edge triggered
P1.6	SCL	Serial port clock line I^2C bus
P1.7	SDA	Serial port data line I^2C bus
P2.0	A8	High order address byte used during external memory accesses
P2.1	A9	
P2.2	A10	
P2.3	A11	
P2.4	A12	
P2.5	A13	
P2.6	A14	
P2.7	A15	
P3.0	RxD	Serial input port (UART)
P3.1	TxD	Serial output port (UART)
P3.2	$\overline{INT0}$	External interrupt 0
P3.3	$\overline{INT1}$	External interrupt 1
P3.4	T0	Timer 0 external input
P3.5	T1	Timer 1 external input
P3.6	$\overline{WR}$	External data memory write strobe
P3.7	$\overline{RD}$	External data memory read strobe
P4.0	CMSR0	Timer T2: compare and set/reset outputs on a match with timer T2
P4.1	CMSR1	
P4.2	CMSR2	
P4.3	CMSR3	
P4.4	CMSR4	
P4.5	CMSR5	Timer T2: compare and toggle outputs on a match with timer T2
P4.6	CMT0	
P4.7	CMT1	
P5.0	ADC0	Eight analogue ADC inputs
P5.1	ADC1	
P5.2	ADC2	
P5.3	ADC3	
P5.4	ADC4	
P5.5	ADC5	
P5.6	ADC6	
P5.7	ADC7	

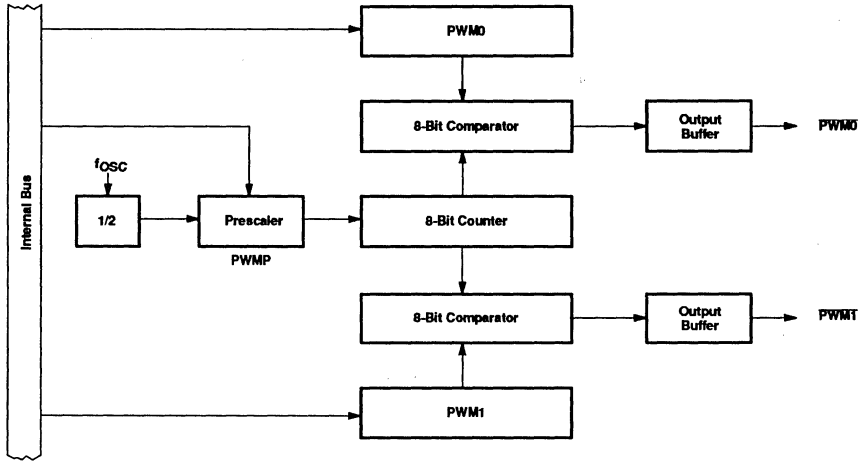


Figure 33. Functional Diagram of Pulse Width Modulated Outputs

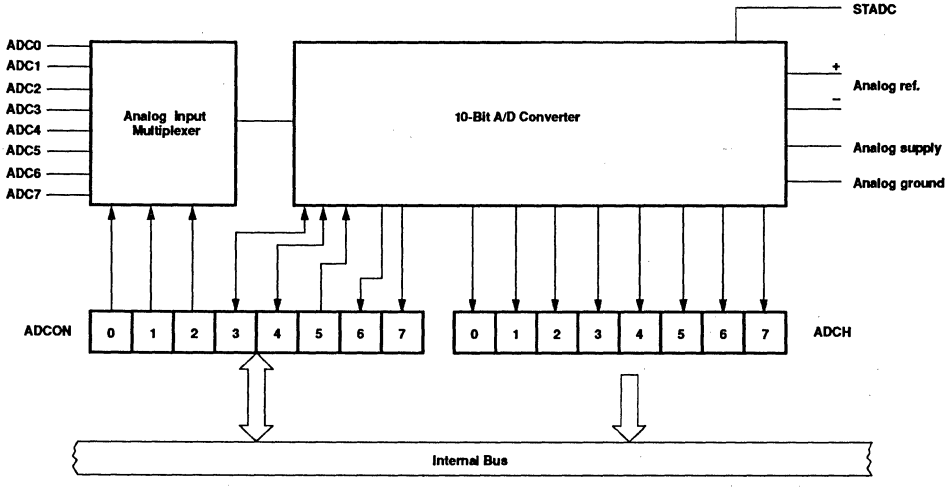


Figure 34. Functional Diagram of Analog Input Circuitry

Analog-to-Digital Conversion: Figure 35 shows the elements of a successive approximation (SA) ADC. The ADC contains a DAC which converts the contents of a successive approximation register to a voltage (VDAC) which is compared to the analog input voltage (Vin). The output of the comparator is fed to the successive approximation control logic which controls the successive approximation register. A conversion is initiated by setting ADCS in the ADCON register. ADCS can be set by software only or by either hardware or software.

The software only start mode is selected when control bit ADCON.5 (ADEX) = 0. A conversion is then started by setting control bit ADCON.3 (ADCS). The hardware or software start mode is selected when ADCON.5 = 1, and a conversion may be started by setting ADCON.3 as above or by applying a rising edge to external pin STADC. When a conversion is started by applying a rising edge, a low level must be applied to STADC for at least one machine cycle followed by a high level for at least one machine cycle.

The low-to-high transition of STADC is recognized at the end of a machine cycle, and the conversion commences at the beginning of the next cycle. When a conversion is initiated by software, the conversion starts at the beginning of the machine cycle which follows the instruction that sets ADCS. ADCS is actually implemented with two flip-flops: a command flip-flop which is affected by set operations, and a status flag which is accessed during read operations.

The next two machine cycles are used to initiate the converter. At the end of the first cycle, the ADCS status flag is set and a value of "1" will be returned if the ADCS flag is read while the conversion is in progress. Sampling of the analog input commences at the end of the second cycle.

During the next eight machine cycles, the voltage at the previously selected pin of port 5 is sampled, and this input voltage should be stable in order to obtain a useful sample.vent, the input voltage slew rate must be less than 10V/ms in order to prevent an undefined result.

The successive approximation control logic first sets the most significant bit and clears all other bits in the successive approximation register (10 0000 0000B). The output of the DAC (50% full scale) is compared to the input voltage Vin. If the input voltage is greater than VDAC, then the bit remains set; otherwise it is cleared.

The successive approximation control logic now sets the next most significant bit (11 0000 0000B or 01 0000 0000B, depending on the previous result), and VDAC is compared to Vin again. If the input voltage is greater than VDAC, then the bit being tested remains set; otherwise the bit being tested is cleared. This process is repeated until all ten bits have been tested, at which stage the result of the conversion is held in the successive approximation register. Figure 36 shows a conversion flow chart. The bit pointer identifies the bit under test. The conversion takes four machine cycles per bit.

The end of the 10-bit conversion is flagged by control bit ADCON.4 (ADCI). The upper 8 bits of the result are held in special function register ADCH, and the two remaining bits are held in ADCON.7 (ADC.1) and ADCON.6 (ADC.0). The user may ignore the two least significant bits in ADCON and use the ADC as an 8-bit converter (8 upper bits in ADCH). In any event, the total actual conversion time is 50 machine cycles. ADCI will be set and the ADCS status flag will be reset 50 cycles after the command flip-flop (ADCS) is set.

Control bits ADCON.0, ADCON.1, and ADCON.2 are used to control an analog multiplexer which selects one of eight analog channels (see Figure 37). An ADC conversion in progress is unaffected by an external or software ADC start. The result of a completed conversion remains unaffected provided ADCI = logic 1; a new ADC conversion already in progress is aborted when the idle or power-down mode is entered. The result of a completed conversion (ADCI = logic 1) remains unaffected when entering the idle mode.

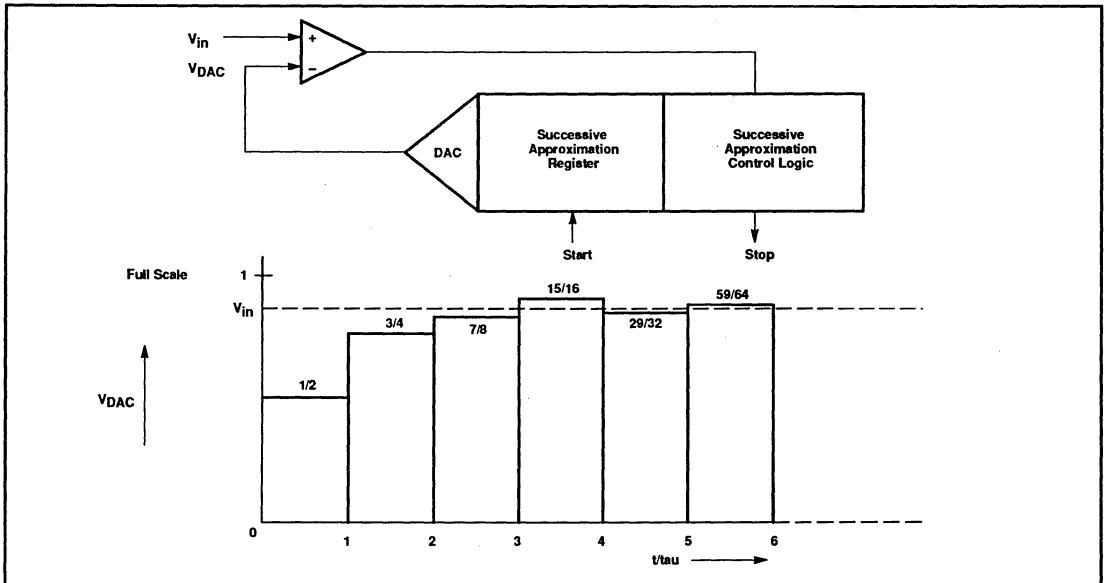


Figure 35. Successive Approximation ADC

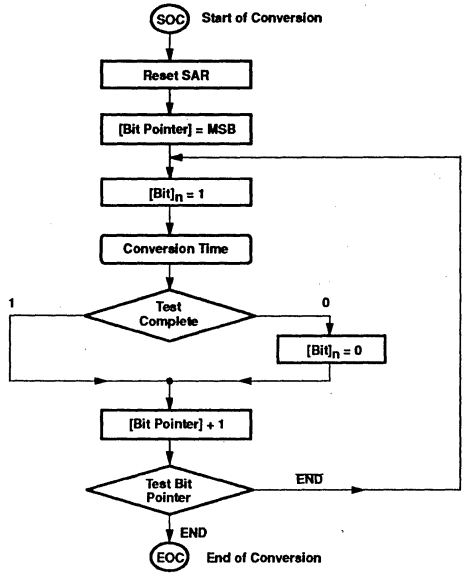


Figure 36. A/D Conversion Flowchart

ADCON (C5H)

7	6	5	4	3	2	1	0
ADC.1	ADC.0	ADEX	ADCI	ADCS	AADR2	AADR1	AADR0
(MSB)							(LSB)

Bit	Symbol	Function
ADCON.7	ADC.1	Bit 1 of ADC result
ADCON.6	ADC.0	Bit 0 of ADC result
ADCON.5	ADEX	Enable external start of conversion by STADC 0 = Conversion can be started by software only (by setting ADCS) 1 = Conversion can be started by software or externally (by a rising edge on STADC)
ADCON.4	ADCI	ADC interrupt flag: this flag is set when an A/D conversion result is ready to be read. An interrupt is invoked if it is enabled. The flag may be cleared by the interrupt service routine. While this flag is set, the ADC cannot start a new conversion. ADCI cannot be set by software.
ADCON.3	ADCS	ADC start and status: setting this bit starts an A/D conversion. It may be set by software or by the external signal STADC. The ADC logic ensures that this signal is HIGH while the ADC is busy. On completion of the conversion, ADCS is reset at the same time the interrupt flag is set. ADCS cannot be reset by software. A new conversion may not be started while either ADCS or ADCI is high.

ADCI	ADCS	ADC Status
0	0	ADC not busy; a conversion can be started
0	1	ADC busy; start of a new conversion is blocked
1	0	Conversion completed; start of a new conversion is blocked
1	1	Not possible

Bit	Symbol	Function
ADCON.2 ADCON.1 ADCON.0	AADR2 AADR1 AADR0	Analogue input select: this binary coded address selects one of the eight analogue port bits of P5 to be input to the converter. It can only be changed when ADCI and ADCS are both LOW.

AADR2	AADR1	AADR0	Selected Analog Channel
0	0	0	ADC0 (P5.0)
0	0	1	ADC1 (P5.1)
0	1	0	ADC2 (P5.2)
0	1	1	ADC3 (P5.3)
1	0	0	ADC4 (P5.4)
0	1	0	ADC5 (P5.5)
1	1	0	ADC6 (P5.6)
1	1	1	ADC7 (P5.7)

Figure 37. ADC Control Register (ADCON)

ADC Resolution and Analog Supply:
Figure 38 shows how the ADC is realized. The ADC has its own supply pins (AV_{DD} and AV_{SS}) and two pins (Vref+ and Vref–) connected to each end of the DAC's resistance-ladder. The ladder has 1023 equally spaced taps, separated by a resistance of "R". The first tap is located 0.5 x R above Vref–, and the last tap is located 1.5 x R below Vref+. This gives a total ladder resistance of 1024 x R. This structure ensures that the DAC is monotonic and results in a symmetrical quantization error as shown in Figure 40.

For input voltages between Vref– and (Vref–) + 1/2 LSB, the 10-bit result of an A/D conversion will be 00 0000 0000B = 000H. For input voltages between (Vref+) – 3/2 LSB and Vref+, the result of a conversion will be 11 1111 1111B = 3FFH. AVref+ and AVref– may be between AV_{DD} + 0.2V and AV_{SS} – 0.2V. AVref+ should be positive with respect to AVref–, and the input voltage (Vin) should be between AVref+ and AVref–. If the analog input voltage range is from 2V to 4V, then 10-bit resolution can be obtained over this range if AVref+ = 4V and AVref– = 2V.

The result can always be calculated from the following formula:

$$\text{Result} = 1024 \times \frac{V_{IN} - AV_{ref-}}{AV_{ref+} - AV_{ref-}}$$

Power Reduction Modes
The 8XC552 has two reduced power modes of operation: the idle mode and the power-down mode. These modes are entered by setting bits in the PCON special function register. When the 8XC552 enters the idle mode, the following functions are disabled:

CPU	(halted)
Timer T2	(halted and reset)
PWM0, PWM1	(reset; outputs are high)
ADC	(conversion aborted if in progress).

In idle mode, the following functions remain active:

Timer 0
Timer 1
Timer T3
SIO0 SIO1
External interrupts

When the 8XC552 enters the power-down mode, the oscillator is stopped. The power-down mode is entered by setting the PD bit in the PCON register. The PD bit can only be set if the $\overline{EW}$ input is tied HIGH.

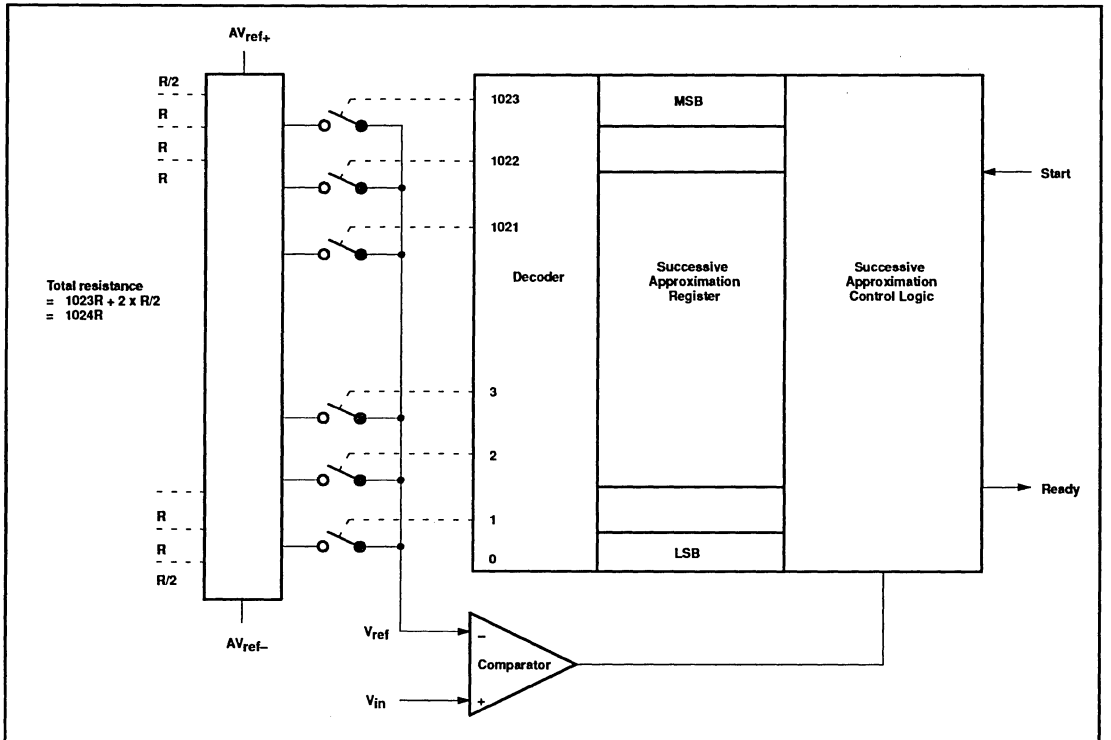
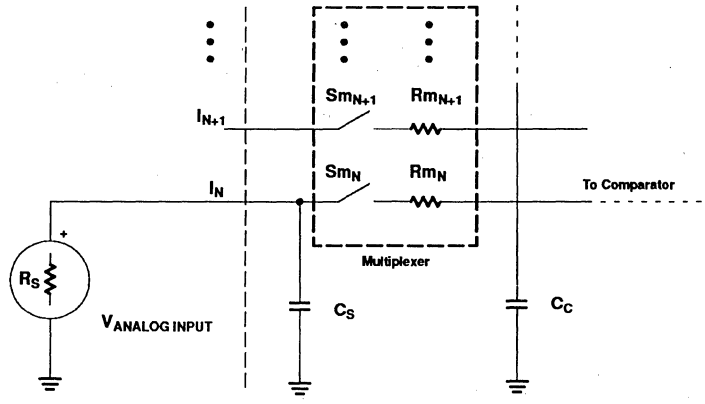


Figure 38. ADC Realization



Rm = 0.5 - 3 kohms
CS + CC = 15pF maximum
RS = Recommended < 9.6 kohms for 1 LSB @ 12MHz

NOTE:
Because the analog to digital converter has a sampled-data comparator, the input looks capacitive to a source. When a conversion is initiated, switch Sm closes for 8tcy (8µs @ 12MHz crystal frequency) during which time capacitance Cs + Cc is charged. It should be noted that the sampling causes the analog input to present a varying load to an analog source.

Figure 39. A/D Input: Equivalent Circuit

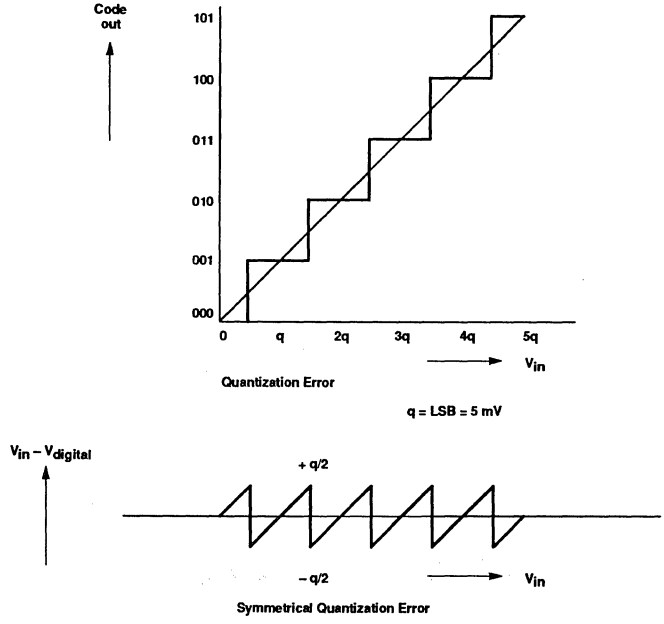


Figure 40. Effective Conversion Characteristic

Power-Down Mode: The instruction that sets PCON.1 will be the last instruction executed in the normal operating mode before the power-down mode is entered. In the power-down mode, the on-chip oscillator is stopped. This freezes all functions; only the on-chip RAM and special function registers are held. The port pins output the contents of their respective special function registers. A hardware reset is the only way to terminate the power-down mode. Reset re-defines all the special function registers, but does not change the on-chip RAM.

In the power-down mode, V_{DD} and AV_{DD} can be reduced to minimize power consumption. V_{DD} and AV_{DD} must not be reduced before the power-down mode is entered and must be restored to the normal operating voltage before the power-down mode is terminated. The reset that terminates the power-down mode also freezes the oscillator. The reset should not be activated before V_{DD} and AV_{DD} are restored to their normal operating level, and must be held active long enough to allow the oscillator to restart and stabilize (normally less than 10ms).

The status of the external pins during power-down is shown in Table 11. If the power-down mode is entered while the 8XC552 is executing out of external program memory, the port data that is held in the P2 special function register is restored to port 2. If a port latch contains a "1", the port pin is held HIGH during the power-down mode by the strong pull-up transistor.

Power Control Register PCON: The idle and power-down modes are entered by writing to bits in PCON. PCON is not bit addressable. See Figure 41.

Memory Organization

The memory organization of the 8XC552 is the same as in the 80C51, with the exception that the 8XC552 has 8k ROM, 256 bytes RAM, and additional SFRs. Addressing modes are the same in the 8XC552 and the 80C51. Details of the differences are given in the following paragraphs.

In the 8XC552, the lower 8k of the 64k program memory address space is filled by internal ROM. By tying the $\overline{EA}$ pin high, the processor fetches instructions from internal program ROM. Bus expansion for accessing program memory from 8k upwards is automatic since external instruction fetches occur automatically when the program counter exceeds 8191. If the $\overline{EA}$ pin is tied low, all program memory fetches are from external memory. The execution speed of the 8XC552 is the same regardless of whether fetches are from external or internal program memory. If all storage is on-chip, then byte location 8191 should be left vacant to prevent an undesired pre-fetch from external program memory address 8192.

Certain locations in program memory are reserved for specific programs. Locations 0000H to 0002H are reserved for the initialization program. Following reset, the CPU always begins execution at locations 0000H. Locations 0003H to 0075H are reserved for the fifteen interrupt request service routines.

Functionally, the internal data memory is the most flexible of the address spaces. The internal data memory space is subdivided into a 256-byte internal data RAM address space and a 128-byte special function register (SFR) address space, as shown in Figure 42.

The internal data RAM address space is 0 to 255. Four 8-bit register banks occupy locations 0 to 31. 128 bit locations of the internal data RAM are accessible through direct addressing. These bits reside in 16 bytes of internal data RAM at locations 20H to 2FH. The stack can be located anywhere in the internal data RAM address space by loading the 8-bit stack pointer. The stack depth may be 256 bytes maximum.

The SFR address space is 128 to 255. All registers except the program counter and the four 8-bit register banks reside in this address space. Memory mapping the SFRs allows them to be accessed as easily as internal RAM, and as such, they can be operated on by most instructions. The 56 SFRs are listed in Figure 43, and their mapping in the SFR address space is shown in Figures 44 and 45. RAM bit addresses are the same as in the 80C51 and are summarized in Figure 46. The special function bit addresses are summarized in Figure 47.

Table 11. External Pin Status During Idle and Power-Down Modes

MODE	MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3	PORT 4	PWM0/PWM1
Idle (1)	Internal	1	1	Port data	Port data	Port data	Port data	Port data	HIGH
Idle (1)	External	1	1	Floating	Port data	Address	Port data	Port data	HIGH
Power-down	Internal	0	0	Port data	Port data	Port data	Port data	Port data	HIGH
Power-down	External	0	0	Floating	Port data	Port data	Port data	Port data	HIGH

	7	6	5	4	3	2	1	0
PCON (87H)	SMOD	–	–	WLE	GF1	GF0	PD	IDL

Bit	Symbol	Function
PCON.7	SMOD	Double Baud rate bit. When set to logic 1 the baud rate is doubled when the serial port SIO0 is being used in modes 1, 2, or 3.
PCON.6	–	(Reserved)
PCON.5	–	(Reserved)
PCON.4	WLE	Watchdog load enable. This flag must be set by software prior to loading timer T3 (watchdog timer). It is cleared when timer T3 is loaded.
PCON.3	GF1	General-purpose flag bit
PCON.2	GF0	General-purpose flag bit
PCON.1	PD	Power-down bit. Setting this bit activates the power-down mode. It can only be set if input EW is high.
PCON.0	IDL	Idle mode bit. Setting this bit activates the idle mode.

If logic 1s are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (0XX00000).

Figure 41. Power Control Register (PCON)

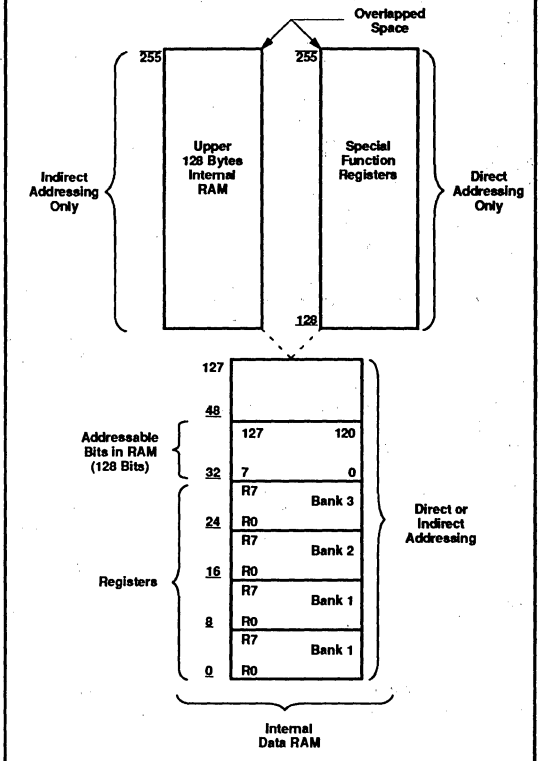


Figure 42. Internal Data Memory Address Space

ARITHMETIC REGISTERS:
ACCumulator,* B register,*
Program Status Word*

POINTERS:
Stack Pointer,
Data Pointer (High and Low)

PARALLEL I/O PORTS:
Port 5,* Port 4,*Port 3,*
Port 2,* Port 1,* Port 0*

INTERRUPT SYSTEM:
Interrupt Priority 0,*
Interrupt Priority 1,*
Interrupt Enable 0,*
Interrupt Enable 1*

PULSE WIDTH MODULATED O/Ps:
Pulse Width Modulation Prescaler
Pulse Width Modulation Register 0,
Pulse Width Modulation Register 1

SERIAL I/O PORTS:
Serial 0 CONtrol,* Serial 0 data BUFfer,
Serial 1 CONtrol,* Serial 1 DATa,
Serial 1 STAtus, Serial 1 ADDress, PCON

TIMERS:
Timer MODe, Timer CONtrol,*
Timer Low 0, Timer High 0,
Timer Low 1, Timer High 1,
TiMer T2 CONtrol, TiMer Low 2,
Timer High 2, Timer T3

CAPTURE AND COMPARE LOGIC:
CapTure CONtrol,
TiMer T2 Interrupt flag Register,
CapTure Low 0, CapTure High 0,
CapTure Low 1, CapTure High 1,
CapTure Low 2, CapTure High 2,
CapTure Low 3, CapTure High 3,
CoMpare Low 0, CoMpare High 0,
CoMpare Low 1, CoMpare High 1,
CoMpare Low 2, CoMpare High 2
SeT Enable, ReseT Enable

ADC
ADC cONtrol, ADC High byte

*NOTE: Bit and byte addressable

Figure 43. Special Function Registers

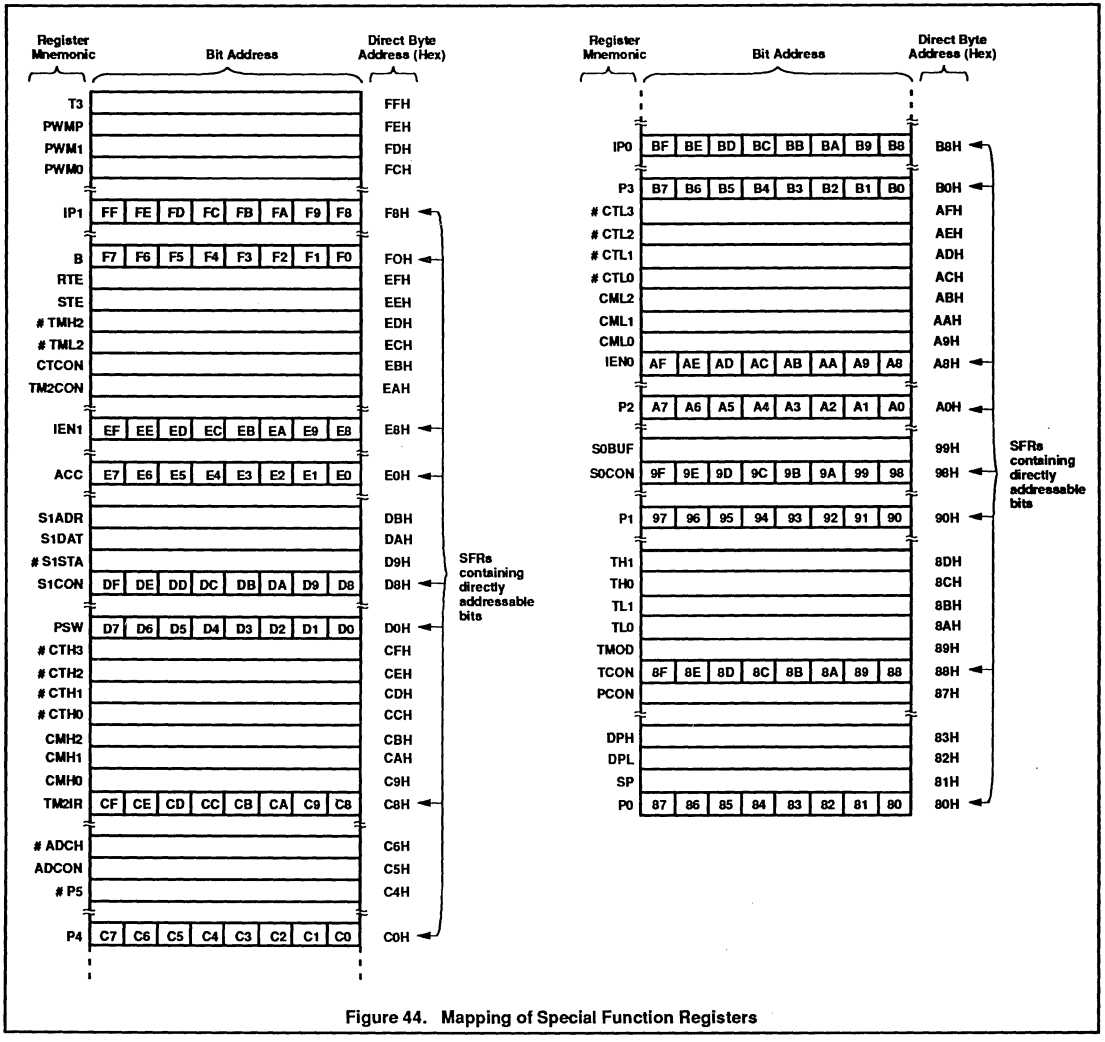


Figure 44. Mapping of Special Function Registers

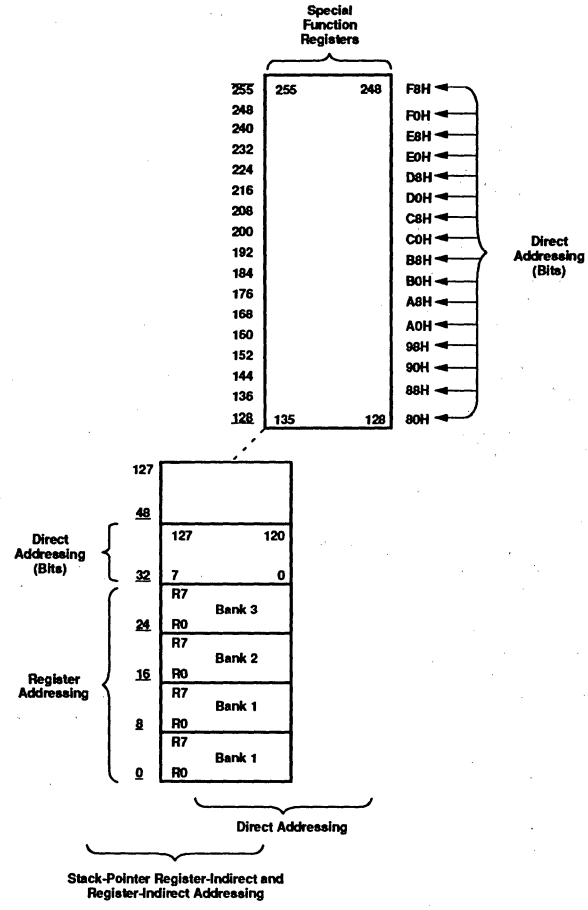


Figure 45. Bit and Byte Addressing Overview of Internal Data Memory

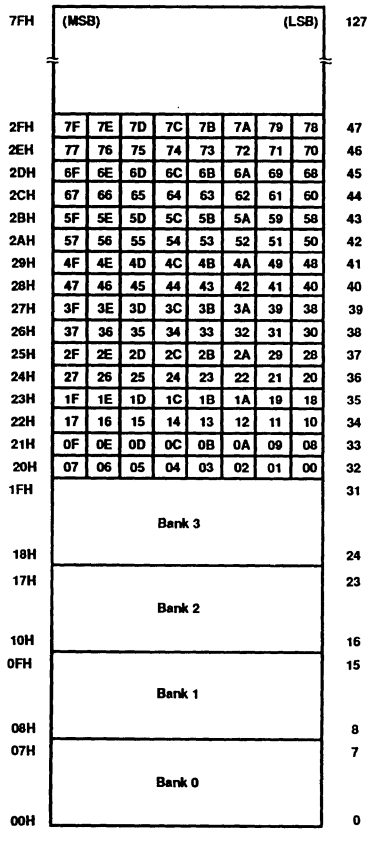


Figure 46. RAM Bit Addresses

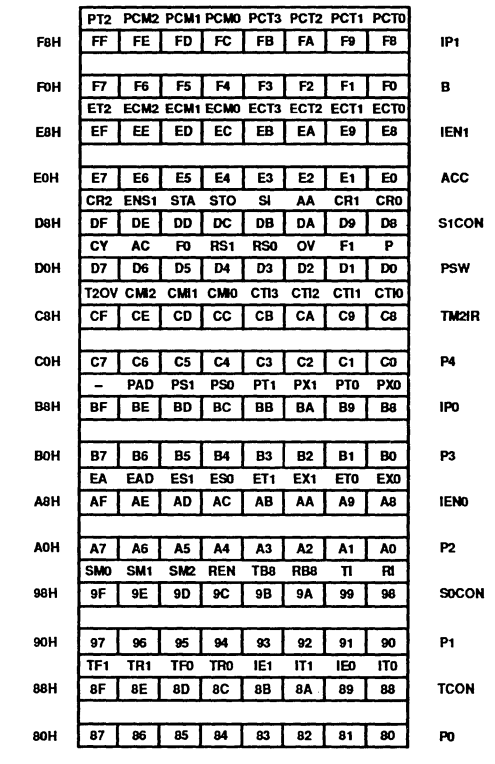


Figure 47. Special Function Register Bit Address

Single-chip 8-bit microcontroller — 80C552/83C552/87C552

Single-chip 8-bit microcontroller with 10-bit A/D, capture/compare timer, high-speed outputs, PWM

DESCRIPTION

The 80C552/83C552/87C552 (hereafter generically referred to as 8XC552) Single-Chip 8-Bit Microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 8XC552 has the same instruction set as the 80C51. Three versions of the derivative exist:

- 83C552 — 8k bytes mask programmable ROM
- 80C552 — ROMless version of the 83C552
- 87C552 — 8k bytes EPROM

The 8XC552 contains a non-volatile 8k × 8 read-only program memory (83C552) EPROM (87C552), a volatile 256 × 8 read/write data memory, five 8-bit I/O ports, one 8-bit input port, two 16-bit timer/event counters (identical to the timers of the 80C51), an additional 16-bit timer coupled to capture and compare latches, a 15-source, two-priority-level, nested interrupt structure, an 8-input ADC, a dual DAC pulse width modulated interface, two serial interfaces (UART and I²C-bus), a "watchdog" timer and on-chip oscillator and timing circuits. For systems that require extra capability, the 8XC552 can be expanded using standard TTL compatible memories and logic.

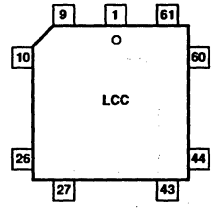
In addition, the 8XC552 has two software selectable modes of power reduction — idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial ports, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

The device also functions as an arithmetic processor having facilities for both binary and BCD arithmetic plus bit-handling capabilities. The instruction set consists of over 100 instructions: 49 one-byte, 45 two-byte, and 17 three-byte. With a 16MHz (24MHz) crystal, 58% of the instructions are executed in 0.75µs (0.5µs) and 40% in 1.5µs (1µs). Multiply and divide instructins require 3µs (2µs).

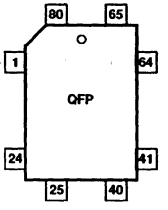
FEATURES

- 80C51 central processing unit
- 8k × 8 ROM expandable externally to 64k bytes
- An additional 16-bit timer/counter coupled to four capture registers and three compare registers
- Two standard 16-bit timer/counters
- 256 × 8 RAM, expandable externally to 64k bytes
- Capable of producing eight synchronized, timed outputs
- A 10-bit ADC with eight multiplexed analog inputs
- Two 8-bit resolution, pulse width modulation outputs
- Five 8-bit I/O ports plus one 8-bit input port shared with analog inputs
- I²C-bus serial I/O port with byte oriented master and slave functions
- Full-duplex UART compatible with the standard 80C51
- On-chip watchdog timer
- Three speed ranges:
  - 16MHz
  - 24MHz
  - 30MHz (in preparation)
- Extended temperature ranges
- OTP package available

PIN CONFIGURATIONS



Pin	Function	Pin	Function
1	P5.0/ADC0	35	XTAL1
2	V_{DD}	36	V_{SS}
3	STADC	37	V_{SS}
4	$\overline{PWM0}$	38	NC
5	$\overline{PWM1}$	39	P2.0/A08
6	$\overline{EW}$	40	P2.1/A09
7	P4.0/CMSR0	41	P2.2/A10
8	P4.1/CMSR1	42	P2.3/A11
9	P4.2/CMSR2	43	P2.4/A12
10	P4.3/CMSR3	44	P2.5/A13
11	P4.4/CMSR4	45	P2.6/A14
12	P4.5/CMSR5	46	P2.7/A15
13	P4.6/CMT0	47	$\overline{PSEN}$
14	P4.7/CMT1	48	ALE/$\overline{PROG}$
15	RST	49	$\overline{EA}$/V_{PP}
16	P1.0/CT0I	50	P0.7/AD7
17	P1.1/CT1I	51	P0.6/AD6
18	P1.2/CT2I	52	P0.5/AD5
19	P1.3/CT3I	53	P0.4/AD4
20	P1.4/T2	54	P0.3/AD3
21	P1.5/RT2	55	P0.2/AD2
22	P1.6/SCL	56	P0.1/AD1
23	P1.7/SDA	57	P0.0/AD0
24	P3.0/RxD	58	AVref–
25	P3.1/TxD	59	AVref+
26	P3.2/$\overline{INT0}$	60	AV_{SS}
27	P3.3/$\overline{INT1}$	61	AV_{DD}
28	P3.4/T0	62	P5.7/ADC7
29	P3.5/T1	63	P5.6/ADC6
30	P3.6/$\overline{WR}$	64	P5.5/ADC5
31	P3.7/$\overline{RD}$	65	P5.4/ADC4
32	NC	66	P5.3/ADC3
33	NC	67	P5.2/ADC2
34	XTAL2	68	P5.1/ADC1



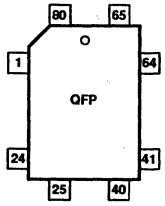
SEE PAGE 424 FOR QFP PIN FUNCTIONS.

PART NUMBER SELECTION

PHILIPS PART ORDER NUMBER PART MARKING		SIGNETICS PART ORDER NUMBER			TEMPERATURE °C AND PACKAGE	FREQ.
ROMless	ROM	ROMless	ROM	EPROM		
PCB80C552-5-16WP	PCB83C552-5WP/xxx	S80C552-1A68	S83C552-1A68	S87C552-4A68	0 to +70, plastic PLCC	16MHz
				S87C552-4K68	0 to +70, ceramic CLCC with window	16MHz
PCB80C552-5-16H	PCB83C552-5H/xxx	S80C552-1B	S83C552-1B	S87C552-4B	0 to +70, plastic QFP	16MHz
PCF80C552-5-16WP	PCF83C552-5WP/xxx	S80C552-2A68	S83C552-2A68	S87C552-5A68	–40 to +85, plastic PLCC	16MHz
				S87C552-5K68	–40 to +85, ceramic CLCC with window	16MHz
PCF80C552-5-16H	PCF83C552-5H/xxx	S80C552-2B	S83C552-2B	S87C552-5B	–40 to +85, plastic QFP	16MHz
PCA80C552-5-16WP	PCA83C552-5WP/xxx	S80C552-6A68	S83C552-6A68		–40 to +125, plastic PLCC	16MHz
PCA80C552-5-16H	PCA83C552-5H/xxx	S80C552-6B	S83C552-6B		–40 to +125, plastic QFP	16MHz
PCB80C552-5-24WP	PCB83C552-5WP/xxx	S80C552-AA68	S83C552-AA68		0 to +70, plastic PLCC	24MHz
PCB80C552-5-24H	PCB83C552-5H/xxx	S80C552-AB	S83C552-AB		0 to +70, plastic QFP	24MHz
PCF80C552-5-24WP	PCF83C552-5WP/xxx	S80C552-BA68	S83C552-BA68		–40 to +85, plastic PLCC	24MHz
PCF80C552-5-24H	PCF83C552-5H/xxx	S80C552-BB	S83C552-BB		–40 to +85, plastic QFP	24MHz

NOTE:
1. xxx denotes the ROM code number.

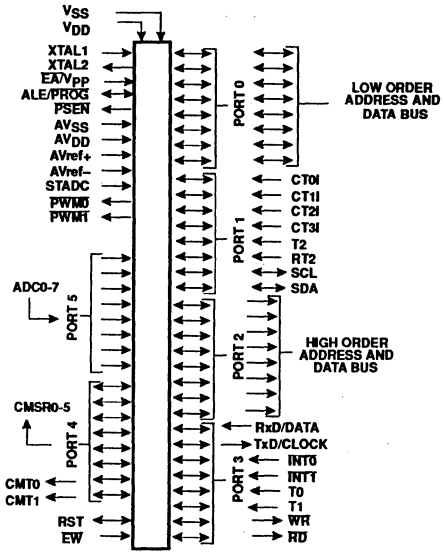
QFP PIN FUNCTIONS



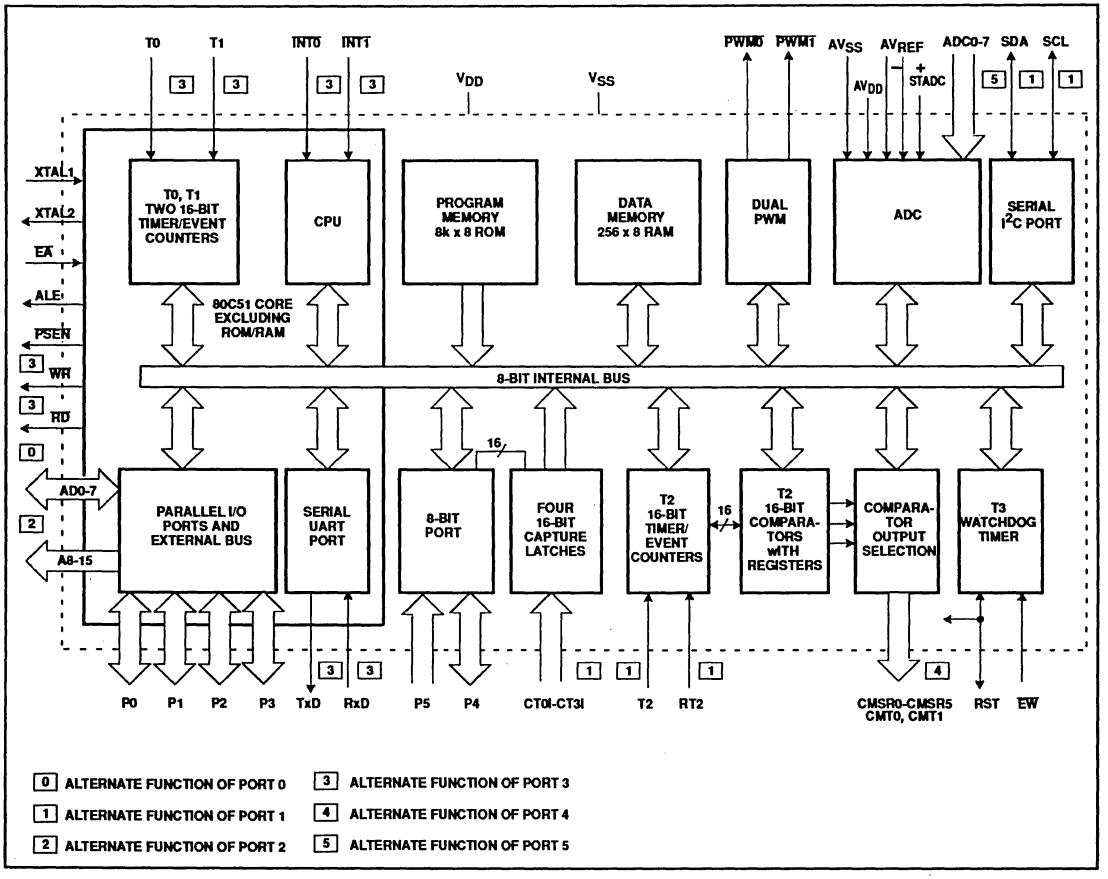
Pin	Function	Pin	Function
1	P4.1/CMSR1	41	P2.3/A11
2	P4.2/CMSR2	42	P2.4/A12
3	NC	43	NC
4	P4.3/CMSR3	44	NC
5	P4.4/CMSR4	45	P2.5/A13
6	P4.5/CMSR5	46	P2.6/A14
7	P4.6/CMT0	47	P2.7/A15
8	P4.7/CMT1	48	$\overline{PSEN}$
9	RST	49	ALE/$\overline{PROG}$
10	P1.0/CT0I	50	$\overline{EA}$/V_{PP}
11	P1.1/CT1I	51	P0.7/AD7
12	P1.2/CT2I	52	P0.6/AD6
13	P1.3/CT3I	53	P0.5/AD5
14	P1.4/T2	54	P0.4/AD4
15	P1.5/RT2	55	P0.3/AD3
16	P1.6/SCL	56	P0.2/AD2
17	P1.7/SDA	57	P0.1/AD1
18	P3.0/RxD	58	P0.0/AD0
19	P3.1/TxD	59	AVref–
20	P3.2/$\overline{INT0}$	60	AVref+
21	NC	61	AV_{SS}
22	NC	62	NC
23	P3.3/$\overline{INT1}$	63	AV_{DD}
24	P3.4/T0	64	P5.7/ADC7
25	P3.5/T1	65	P5.6/ADC6
26	P3.6/$\overline{WR}$	66	P5.5/ADC5
27	P3.7/$\overline{RD}$	67	P5.4/ADC4
28	NC	68	P5.3/ADC3
29	NC	69	P5.2/ADC2
30	NC	70	P5.1/ADC1
31	XTAL2	71	P5.0/ADC0
32	XTAL1	72	V_{DD}
33	IC	73	IC
34	V_{SS}	74	STADC
35	V_{SS}	75	$\overline{PWM0}$
36	V_{SS}	76	$\overline{PWM1}$
37	NC	77	$\overline{EW}$
38	P2.0/A08	78	NC
39	P2.1/A09	79	NC
40	P2.2/A10	80	P4.0/CMSR0

NC = not connected
IC = internally connected (do not use)

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. PLCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{DD}	2	72	I	**Digital Power Supply:** +5V power supply pin during normal operation, idle and power-down mode.
STADC	3	74	I	**Start ADC Operation:** Input starting analog to digital conversion (ADC operation can also be started by software).
$\overline{PWM0}$	4	75	O	**Pulse Width Modulation:** Output 0.
$\overline{PWM1}$	5	76	O	**Pulse Width Modulation:** Output 1.
$\overline{EW}$	6	77	I	**Enable Watchdog Timer:** Enable for T3 watchdog timer and disable power-down mode.
P0.0-P0.7	57-50	58-51	I/O	**Port 0:** Port 0 is an 8-bit open-drain bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application it uses strong internal pull-ups when emitting 1s. Port 0 is also used to input the code byte during programming and to output the code byte during verification.
P1.0-P1.7	16-23	10-17	I/O	**Port 1:** 8-bit I/O port. Alternate functions include:
	16-21	10-15	I/O	**(P1.0-P1.5):** Quasi-bidirectional port pins.
	22-23	16-17	I/O	**(P1.6, P1.7):** Open drain port pins.
	16-19	10-13	I	**CT0I-CT3I (P1.0-P1.3):** Capture timer input signals for timer T2.
	20	14	I	**T2 (P1.4):** T2 event input.
	21	15	I	**RT2 (P1.5):** T2 timer reset signal. Rising edge triggered.
	22	16	I/O	**SCL (P1.6):** Serial port clock line I^2C-bus.
	23	17	I/O	**SDA (P1.7):** Serial port data line I^2C-bus.
				Port 1 is also used to input the lower order address byte during EPROM programming and verification. A0 is on P1.0, etc.
P2.0-P2.7	39-46	38-42, 45-47	I/O	**Port 2:** 8-bit quasi-bidirectional I/O port. Alternate function: High-order address byte for external memory (A08-A15). Port 2 is also used to input the upper order address during EPROM programming and verification. A8 is on P2.0, A9 on P2.1, through A12 on P2.4.
P3.0-P3.7	24-31	18-20, 23-27	I/O	**Port 3:** 8-bit quasi-bidirectional I/O port. Alternate functions include:
	24	18		**RxD(P3.0):** Serial input port.
	25	19		**TxD (P3.1):** Serial output port.
	26	20		**$\overline{INT0}$ (P3.2):** External interrupt.
	27	23		**$\overline{INT1}$ (P3.3):** External interrupt.
	28	24		**T0 (P3.4):** Timer 0 external input.
	29	25		**T1 (P3.5):** Timer 1 external input.
	30	26		**$\overline{WR}$ (P3.6):** External data memory write strobe.
	31	27		**$\overline{RD}$ (P3.7):** External data memory read strobe.
P4.0-P4.7	7-14	80, 1-2 4-8	I/O	**Port 4:** 8-bit quasi-bidirectional I/O port. Alternate functions include:
	7-12	80, 1-2 4-6	O	**CMSR0-CMSR5 (P4.0-P4.5):** Timer T2 compare and set/reset outputs on a match with timer T2.
	13, 14	7, 8	O	**CMT0, CMT1 (P4.6, P4.7):** Timer T2 compare and toggle outputs on a match with timer T2.
P5.0-P5.7	68-62, 1	71-64,	I	**Port 5:** 8-bit input port. **ADC0-ADC7 (P5.0-P5.7):** Alternate function: Eight input channels to ADC.
RST	15	9	I/O	**Reset:** Input to reset the 8XC552. It also provides a reset pulse as output when timer T3 overflows.
XTAL1	35	32	I	**Crystal Input 1:** Input to the inverting amplifier that forms the oscillator, and input to the internal clock generator. Receives the external clock signal when an external oscillator is used.
XTAL2	34	31	O	**Crystal Input 2:** Output of the inverting amplifier that forms the oscillator. Left open-circuit when an external clock is used.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO. PLCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{SS}	36, 37	34-36	I	**Digital ground.**
$\overline{PSEN}$	47	48	O	**Program Store Enable:** Active-low read strobe to external program memory.
ALE/$\overline{PROG}$	48	49	O	**Address Latch Enable:** Latches the low byte of the address during accesses to external memory. It is activated every six oscillator periods. During an external data memory access, one ALE pulse is skipped. ALE can drive up to eight LS TTL inputs and handles CMOS inputs without an external pull-up. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{EA}$/V_{PP}	49	50	I	**External Access:** When $\overline{EA}$ is held at TTL level high, the CPU executes out of the internal program ROM provided the program counter is less than 8192. When $\overline{EA}$ is held at TTL low level, the CPU executes out of external program memory. $\overline{EA}$ is not allowed to float. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
AV_{REF-}	58	59	I	**Analog to Digital Conversion Reference Resistor:** Low-end.
AV_{REF+}	59	60	I	**Analog to Digital Conversion Reference Resistor:** High-end.
AV_{SS}	60	61	I	**Analog Ground**
AV_{DD}	61	63	I	**Analog Power Supply**

NOTE:
1. To avoid "latch-up" effect at power-on, the voltage on any pin at any time must not be higher or lower than V_{DD} + 0.5V or V_{SS} – 0.5V, respectively.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 424.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-on reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-on, the voltage on V_{DD} and RST must come up at the same time for a proper start-up.

IDLE MODE

In the idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. The control bits for the reduced power modes are in the special function register PCON. Table 1 shows the state of the I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	PSEN	PORT 0	PORT 1	PORT 2	PORT 3	PORT 4	PWM0/ PWM1
Idle	Internal	1	1	Data	Data	Data	Data	Data	High
Idle	External	1	1	Float	Data	Address	Data	Data	High
Power-down	Internal	0	0	Data	Data	Data	Data	Data	High
Power-down	External	0	0	Float	Data	Data	Data	Data	High

Serial Control Register (S1CON) – See Table 2

S1CON (D8H)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

Bits CR0, CR1 and CR2 determine the serial clock frequency that is generated in the master mode of operation.

Table 2. Serial Clock Rates

CR2	CR1	CR0	BIT FREQUENCY (kHz) AT f_{OSC} 6MHz	12MHz	16MHz	24MHz[2]	f_{OSC} DIVIDED BY
0	0	0	23	47	62.5	94	256
0	0	1	27	54	71	107 [1]	224
0	1	0	31.25	62.5	83.3	125 [1]	192
0	1	1	37	75	100	150 [1]	160
1	0	0	6.25	12.5	17	25	960
1	0	1	50	100	133 [1]	200 [1]	120
1	1	0	100	200	267 [1]	400 [1]	60
1	1	1	0.25 < 62.5	0.5 < 62.5	0.67 < 56	0.98 < 50	96 × (256 – (reload value Timer 1)) Timer 1 in Mode 2.

NOTE:
1. These frequencies exceed the upper limit of 100kHz of the I²C-bus specification and cannot be used in an I²C-bus application.
2. At f_{OSC} = 24MHz the maximum I²C bus rate of 100kHz cannot be realized due to the fixed divider rates. For f_{OSC} = 24MHz the maximum rate is limited to 94kHz.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ to V_{SS} (87C552 only)	–0.5 to +13	V
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Input, output DC current on any single I/O pin	5.0	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.0	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DEVICE SPECIFICATIONS

TYPE	SUPPLY VOLTAGE (V) MIN	MAX	FREQUENCY (MHz) MIN	MAX	TEMPERATURE RANGE (°C)
PCB83(0)C552-5-16	4.0	6.0	1.2	16	0 to +70
P87C552-4	4.5	5.5	3.5	16	0 to +70
PCF83(0)C552-5-16	4.0	6.0	1.2	16	–40 to +85
P87C552-5	4.5	5.5	3.5	16	–40 to +85
PCA83(0)C552-5-16	4.5	5.5	1.2	16	–40 to +125
PCB83(0)C552-5-24	4.5	5.5	1.2	24	0 to +70
PCF83(0)C552-5-24	4.5	5.5	1.2	24	–40 to +85

DC ELECTRICAL CHARACTERISTICS

V_{SS}, AV_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
I_{DD}	Supply current operating:	See notes 1 and 2			
	PCB8XC552-5-16	f_{OSC} = 16MHz		45	mA
	PCF8XC552-5-16	f_{OSC} = 16MHz		45	mA
	PCA8XC552-5-16	f_{OSC} = 16MHz		40	mA
	87C552	f_{OSC} = 16MHz		35	mA
	PCB8XC552-5-24	f_{OSC} = 24MHz		55	mA
	PCF8XC552-5-24	f_{OSC} = 24MHz		55	mA
I_{ID}	Idle mode:	See notes 1 and 3			
	PCB8XC552-5-16	f_{OSC} = 16MHz		10	mA
	PCF8XC552-5-16	f_{OSC} = 16MHz		10	mA
	PCA8XC552-5-16	f_{OSC} = 16MHz		9	mA
	87C552	f_{OSC} = 16MHz		7	mA
	PCB8XC552-5-24	f_{OSC} = 24MHz		12.5	mA
	PCF8XC552-5-24	f_{OSC} = 24MHz		12.5	mA
I_{PD}	Power-down current:	See notes 1 and 4; 2V < V_{PD} < V_{DD} max			
	PCB8XC552			50	µA
	PCF8XC552			50	µA
	PCA8XC552			150	µA
	87C552			50	µA
Inputs					
V_{IL}	Input low voltage, except $\overline{EA}$, P1.6, P1.7		–0.5	$0.2V_{DD}$–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$		–0.5	$0.2V_{DD}$–0.3	V
V_{IL2}	Input low voltage to P1.6/SCL, P1.7/SDA[5]		–0.5	$0.3V_{DD}$	V
V_{IH}	Input high voltage, except XTAL1, RST		$0.2V_{DD}$+0.9	V_{DD}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST		$0.7V_{DD}$	V_{DD}+0.5	V
V_{IH2}	Input high voltage, P1.6/SCL, P1.7/SDA[5]		$0.7V_{DD}$	6.0	V
I_{IL}	Logical 0 input current, ports 1, 2, 3, 4, except P1.6, P1.7	V_{IN} = 0.45V		–50	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3, 4, except P1.6, P1.7	See note 6		–650	µA
$\pm I_{IL1}$	Input leakage current, port 0, $\overline{EA}$, STADC, $\overline{EW}$	0.45V < V_I < V_{DD}		10	µA
$\pm I_{IL2}$	Input leakage current, P1.6/SCL, P1.7/SDA	0V < V_I < 6V 0V < V_{DD} < 5.5V		10	µA
$\pm I_{IL3}$	Input leakage current, port 5	0.45V < V_I < V_{DD}		1	µA
Outputs					
V_{OL}	Output low voltage, ports 1, 2, 3, 4, except P1.6, P1.7	I_{OL} = 1.6mA[7]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$, $\overline{PWM0}$, $\overline{PWM1}$	I_{OL} = 3.2mA[7]		0.45	V
V_{OL2}	Output low voltage, P1.6/SCL, P1.7/SDA	I_{OL} = 3.0mA[7]		0.4	V
V_{OH}	Output high voltage, ports 1, 2, 3, 4, except P1.6/SCL, P1.7/SDA				
		$-I_{OH}$ = 60µA	2.4		V
		$-I_{OH}$ = 25µA	$0.75V_{DD}$		V
		$-I_{OH}$ = 10µA	$0.9V_{DD}$		V
V_{OH1}	Output high voltage (port 0 in external bus mode, ALE, $\overline{PSEN}$, $\overline{PWM0}$, $\overline{PWM1}$)[8]				
		$-I_{OH}$ = 400µA	2.4		V
		$-I_{OH}$ = 150µA	$0.75V_{DD}$		V
		$-I_{OH}$ = 40µA	$0.9V_{DD}$		V
V_{OH2}	Output high voltage (RST)	$-I_{OH}$ = 400µA	2.4		V
		$-I_{OH}$ = 120µA	$0.8V_{DD}$		V

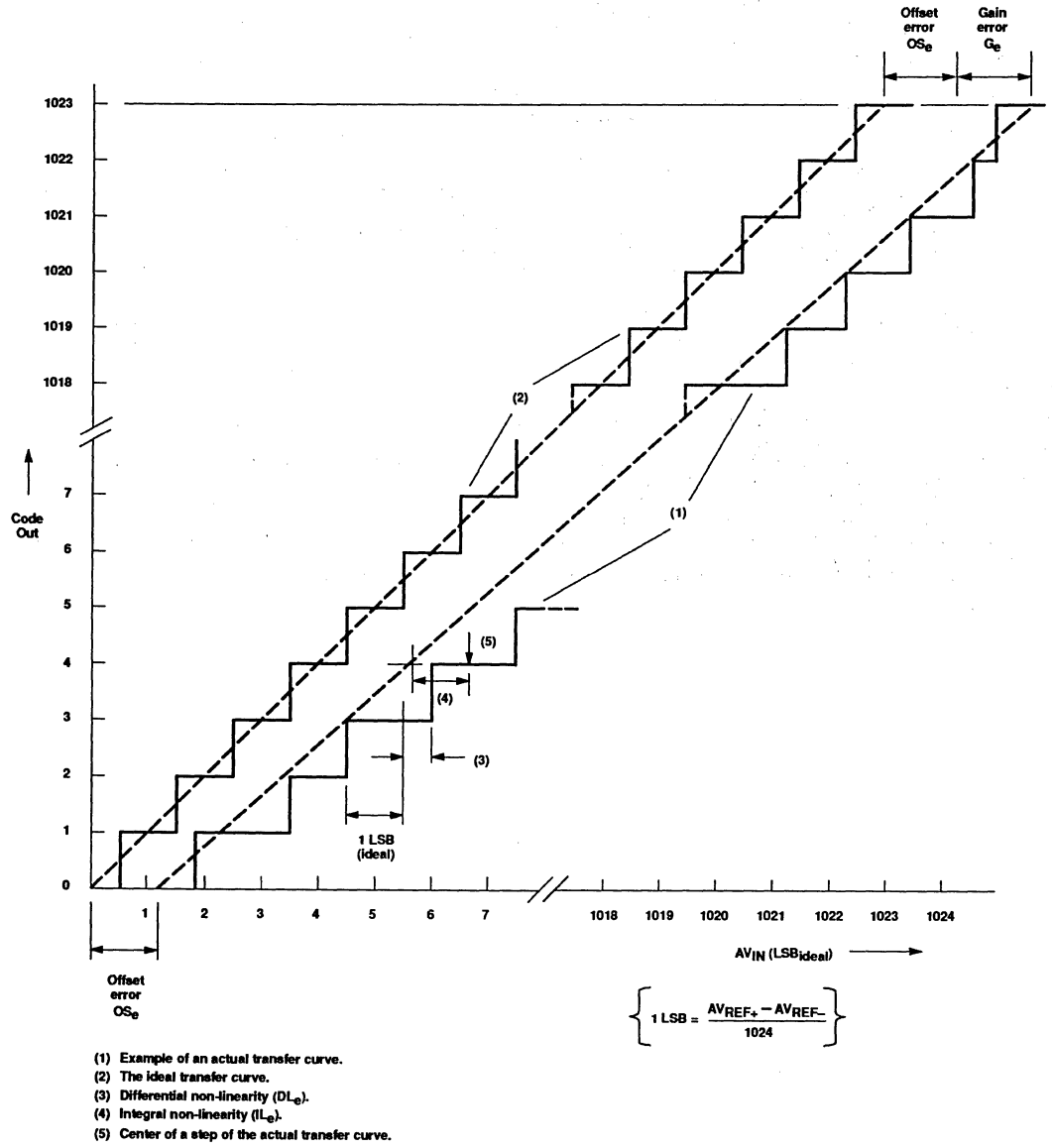
DC ELECTRICAL CHARACTERISTICS (Continued)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
Outputs (Continued)					
R_{RST}	Internal reset pull-down resistor		50	150	kΩ
C_{IO}	Pin capacitance	Test freq = 1MHz, T_{amb} = 25°C		10	pF
Analog Inputs					
AV_{DD}	Analog supply voltage:				
	PCB8XC552-5-16	$AV_{DD} = V_{DD} \pm 0.2V$	4.0	6.0	V
	PCF8XC552-5-16	$AV_{DD} = V_{DD} \pm 0.2V$	4.0	6.0	V
	PCA8XC552-5-16	$AV_{DD} = V_{DD} \pm 0.2V$	4.5	5.5	V
	87C552[9]	$AV_{DD} = V_{DD} \pm 0.2V$	4.5	5.5	V
	PCB8XC552-5-24	$AV_{DD} = V_{DD} \pm 0.2V$	4.5	5.5	V
	PCF8XC552-5-24	$AV_{DD} = V_{DD} \pm 0.2V$	4.5	5.5	V
AI_{DD}	Analog supply current: operating: (16MHz)	Port 5 = 0 to AV_{DD}		1.2	mA
	Analog supply current: operating: (24MHz)	Port 5 = 0 to AV_{DD}		1.0	mA
AI_{ID}	Idle mode:				
	PCB8XC552-5-16			50	µA
	PCF8XC552-5-16			50	µA
	PCA8XC552-5-16			100	µA
	87C552			50	µA
	PCB8XC552-5-24			50	µA
	PCF8XC552-5-24			50	µA
AI_{PD}	Power-down mode:	$2V < AV_{PD} < AV_{DD}$ max			
	PCB8XC552			50	µA
	PCF8XC552			50	µA
	PCA8XC552			100	µA
	87C552			50	µA
AV_{IN}	Analog input voltage		$AV_{SS}-0.2$	$AV_{DD}+0.2$	V
AV_{REF}	Reference voltage:				
	AV_{REF-}		$AV_{SS}-0.2$		V
	AV_{REF+}			$AV_{DD}+0.2$	V
R_{REF}	Resistance between AV_{REF+} and AV_{REF-}		10	50	kΩ
C_{IA}	Analog input capacitance			15	pF
t_{ADS}	Sampling time			$8t_{CY}$	µs
t_{ADC}	Conversion time (including sampling time)			$50t_{CY}$	µs
DL_e	Differential non-linearity[10, 11, 12]			±1	LSB
IL_e	Integral non-linearity[10, 13]			±2	LSB
OS_e	Offset error[10, 14]			±2	LSB
G_e	Gain error[10, 15]			±0.4	%
A_e	Absolute voltage error[10, 16]			±3	LSB
M_{CTC}	Channel to channel matching			±1	LSB
C_t	Crosstalk between inputs of port 5[17]	0–100kHz		–60	dB

NOTES: See Next Page.

NOTES FOR DC ELECTRICAL CHARACTERISTICS:

1. See Figures 10 through 15 for I_{DD} test conditions.
2. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = $\overline{EW}$ = V_{DD}; STADC = V_{SS}.
3. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; XTAL2 not connected; Port 0 = $\overline{EW}$ = V_{DD}; $\overline{EA}$ = RST = STADC = V_{SS}.
4. The power-down current is measured with all output pins disconnected; XTAL2 not connected; Port 0 = $\overline{EW}$ = V_{DD}; $\overline{EA}$ = RST = STADC = XTAL1 = V_{SS}.
5. The input threshhold voltage of P1.6 and P1.7 (SIO1) meets the I^2C specification, so an input voltage below 1.5V will be recognized as a logic 0 while an input voltage above 3.0V will be recognized as a logic 1.
6. Pins of ports 1 (except P1.6, P1.7), 2, 3, and 4 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
7. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. I_{OL} can exceed these conditions provided that no single output sinks more than 5mA and no more than two outputs exceed the test conditions.
8. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{DD}$ specification when the address bits are stabilizing.
9. The following condition must not be exceeded: $V_{DD} - 0.2V < AV_{DD} < V_{DD} + 0.2V$.
10. Conditions: AV_{REF-} = 0V; AV_{DD} = 5.0V, AV_{REF+} (80C552, 83C552) = 5.12V. ADC is monotonic with no missing codes. Measurement by continuous conversion of AV_{IN} = –20mV to 5.12V in steps of 0.5mV, derivating parameters from collected conversion results of ADC. AV_{REF+} (87C552) = 4.977V. ADC is monotonic with no missing codes.
11. The differential non-linearity (DL_e) is the difference between the actual step width and the ideal step width. (See Figure 1.)
12. The ADC is monotonic; there are no missing codes.
13. The integral non-linearity (IL_e) is the peak difference between the center of the steps of the actual and the ideal transfer curve after appropriate adjustment of gain and offset error. (See Figure 1.)
14. The offset error (OS_e) is the absolute difference between the straight line which fits the actual transfer curve (after removing gain error), and a straight line which fits the ideal transfer curve. (See Figure 1.)
15. The gain error (G_e) is the relative difference in percent between the straight line fitting the actual transfer curve (after removing offset error), and the straight line which fits the ideal transfer curve. Gain error is constant at every point on the transfer curve. (See Figure 1.)
16. The absolute voltage error (A_e) is the maximum difference between the center of the steps of the actual transfer curve of the non-calibrated ADC and the ideal transfer curve.
17. This should be considered when both analog and digital signals are simultaneously input to port 5.



(1) Example of an actual transfer curve.
(2) The ideal transfer curve.
(3) Differential non-linearity (DL_e).
(4) Integral non-linearity (IL_e).
(5) Center of a step of the actual transfer curve.

Figure 1. ADC Conversion Characteristic

AC ELECTRICAL CHARACTERISTICS[1, 2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		16MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	2	Oscillator frequency[3]					1.2	16	MHz
t_{LHLL}	2	ALE pulse width	127		85		$2t_{CLCL}-40$		ns
t_{AVLL}	2	Address valid to ALE low	28		8		$t_{CLCL}-55$		ns
t_{LLAX}	2	Address hold after ALE low	48		28		$t_{CLCL}-35$		ns
t_{LLIV}	2	ALE low to valid instruction in		234		150		$4t_{CLCL}-100$	ns
t_{LLPL}	2	ALE low to $\overline{PSEN}$ low	43		23		$t_{CLCL}-40$		ns
t_{PLPH}	2	$\overline{PSEN}$ pulse width	205		143		$3t_{CLCL}-45$		ns
t_{PLIV}	2	$\overline{PSEN}$ low to valid instruction in		145		83		$3t_{CLCL}-105$	ns
t_{PXIX}	2	Input instruction hold after $\overline{PSEN}$	0		0		0		ns
t_{PXIZ}	2	Input instruction float after $\overline{PSEN}$		59		38		$t_{CLCL}-25$	ns
t_{AVIV}	2	Address to valid instruction in		312		208		$5t_{CLCL}-105$	ns
t_{PLAZ}	2	$\overline{PSEN}$ low to address float		10		10		10	ns
Data Memory									
t_{AVLL}	3, 4	Address valid to ALE low	43		23		$t_{CLCL}-40$		ns
t_{RLRH}	3	$\overline{RD}$ pulse width	400		275		$6t_{CLCL}-100$		ns
t_{WLWH}	4	$\overline{WR}$ pulse width	400		275		$6t_{CLCL}-100$		ns
t_{RLDV}	3	$\overline{RD}$ low to valid data in		252		148		$5t_{CLCL}-165$	ns
t_{RHDX}	3	Data hold after $\overline{RD}$	0		0		0		ns
t_{RHDZ}	3	Data float after $\overline{RD}$		97		55		$2t_{CLCL}-70$	ns
t_{LLDV}	3	ALE low to valid data in		517		350		$8t_{CLCL}-150$	ns
t_{AVDV}	3	Address to valid data in		585		398		$9t_{CLCL}-165$	ns
t_{LLWL}	3, 4	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	138	238	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	3, 4	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		120		$4t_{CLCL}-130$		ns
t_{QVWX}	4	Data valid to $\overline{WR}$ transition	23		3		$t_{CLCL}-60$		ns
t_{DW}	4	Data before $\overline{WR}$	433		288		$7t_{CLCL}-150$		ns
t_{WHQX}	4	Data hold after $\overline{WR}$	33		13		$t_{CLCL}-50$		ns
t_{RLAZ}	3	$\overline{RD}$ low to address float		0		0		0	ns
t_{WHLH}	3, 4	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	23	103	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock									
t_{CHCX}	5	High time[4]	20		20		20		ns
t_{CLCX}	5	Low time[4]	20		20		20		ns
t_{CLCH}	5	Rise time[4]		20		20		20	ns
t_{CHCL}	5	Fall time[4]		20		20		20	ns
Serial Timing – Shift Register Mode[4] (Test Conditions: T_{amb} = 0°C to +70°C; V_{SS} = 0V; Load Capaciatnce = 80pF)									
t_{XLXL}	6	Serial port clock cycle time	1.0		0.75		$12t_{CLCL}$		µs
t_{QVXH}	6	Output data setup to clock rising edge	700		492		$10t_{CLCL}-133$		ns
t_{XHQX}	6	Output data hold after clock rising edge	50		8		$2t_{CLCL}-117$		ns
t_{XHDX}	6	Input data hold after clock rising edge	0		0		0		ns
t_{XHDV}	6	Clock rising edge to input data valid		700		492		$10t_{CLCL}-133$	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. 87C552: $1/t_{CLCL}$ = 3.5 to 16 MHz.
4. These values are characterized but not 100% production tested.
5. $t_{CLCL} = 1/f_{OSC}$ = one oscillator clock period.
   t_{CLCL} = 83.3ns at f_{OSC} = 12MHz.
   t_{CLCL} = 62.5ns at f_{OSC} = 16MHz.

AC ELECTRICAL CHARACTERISTICS (Continued)[1, 2]

SYMBOL	FIGURE	PARAMETER	24MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	2	Oscillator frequency			1.2	24	MHz
t_{LHLL}	2	ALE pulse width	43		$2t_{CLCL}-40$		ns
t_{AVLL}	2	Address valid to ALE low	17		$t_{CLCL}-25$		ns
t_{LLAX}	2	Address hold after ALE low	17		$t_{CLCL}-25$		ns
t_{LLIV}	2	ALE low to valid instruction in		102		$4t_{CLCL}-65$	ns
t_{LLPL}	2	ALE low to $\overline{PSEN}$ low	17		$t_{CLCL}-25$		ns
t_{PLPH}	2	$\overline{PSEN}$ pulse width	80		$3t_{CLCL}-45$		ns
t_{PLIV}	2	$\overline{PSEN}$ low to valid instruction in		65		$3t_{CLCL}-60$	ns
t_{PXIX}	2	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	2	Input instruction float after $\overline{PSEN}$		17		$t_{CLCL}-25$	ns
t_{AVIV}	2	Address to valid instruction in		128		$5t_{CLCL}-80$	ns
t_{PLAZ}	2	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{AVLL}	3, 4	Address valid to ALE low	17		$t_{CLCL}-25$		ns
t_{RLRH}	3	$\overline{RD}$ pulse width	150		$6t_{CLCL}-100$		ns
t_{WLWH}	4	$\overline{WR}$ pulse width	150		$6t_{CLCL}-100$		ns
t_{RLDV}	3	$\overline{RD}$ low to valid data in		118		$5t_{CLCL}-90$	ns
t_{RHDX}	3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	3	Data float after $\overline{RD}$		55		$2t_{CLCL}-28$	ns
t_{LLDV}	3	ALE low to valid data in		183		$8t_{CLCL}-150$	ns
t_{AVDV}	3	Address to valid data in		210		$9t_{CLCL}-165$	ns
t_{LLWL}	3, 4	ALE low to $\overline{RD}$ or $\overline{WR}$ low	75	175	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	3, 4	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	92		$4t_{CLCL}-75$		ns
t_{QVWX}	4	Data valid to $\overline{WR}$ transition	12		$t_{CLCL}-30$		ns
t_{DW}	4	Data before $\overline{WR}$	162		$7t_{CLCL}-130$		ns
t_{WHQX}	4	Data hold after $\overline{WR}$	17		$t_{CLCL}-25$		ns
t_{RLAZ}	3	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	3, 4	$\overline{RD}$ or $\overline{WR}$ high to ALE high	17	67	$t_{CLCL}-25$	$t_{CLCL}+25$	ns
External Clock							
t_{CHCX}	5	High time[3]	17		17		ns
t_{CLCX}	5	Low time[3]	17		17		ns
t_{CLCH}	5	Rise time[3]		20		20	ns
t_{CHCL}	5	Fall time[3]		20		20	ns
Serial Timing – Shift Register Mode[3] (Test Conditions: T_{amb} = 0°C to +70°C; V_{SS} = 0V; Load Capaciatnce = 80pF)							
t_{XLXL}	6	Serial port clock cycle time	0.5		$12t_{CLCL}$		µs
t_{QVXH}	6	Output data setup to clock rising edge	283		$10t_{CLCL}-133$		ns
t_{XHQX}	6	Output data hold after clock rising edge	23		$2t_{CLCL}-60$		ns
t_{XHDX}	6	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	6	Clock rising edge to input data valid		283		$10t_{CLCL}-133$	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. These values are characterized but not 100% production tested.
4. $t_{CLCL} = 1/f_{OSC}$ = one oscillator clock period.
   t_{CLCL} = 41.7ns at f_{OSC} = 24MHz.

AC ELECTRICAL CHARACTERISTICS (Continued)

SYMBOL	PARAMETER	INPUT	OUTPUT
I²C Interface (Refer to Figure 9)			
$t_{HD;STA}$	START condition hold time	≥ 14 t_{CLCL}	> 4.0µs [1]
t_{LOW}	SCL low time	≥ 16 t_{CLCL}	> 4.7µs [1]
t_{HIGH}	SCL high time	≥ 14 t_{CLCL}	> 4.0µs [1]
t_{RC}	SCL rise time	≤ 1µs	– [2]
t_{FC}	SCL fall time	≤ 0.3µs	< 0.3µs [3]
$t_{SU;DAT1}$	Data set-up time	≥ 250ns	> 20 $t_{CLCL} - t_{RD}$
$t_{SU;DAT2}$	SDA set-up time (before rep. START cond.)	≥ 250ns	> 1µs [1]
$t_{SU;DAT3}$	SDA set-up time (before STOP cond.)	≥ 250ns	> 8 t_{CLCL}
$t_{HD;DAT}$	Data hold time	≥ 0ns	> 8 $t_{CLCL} - t_{FC}$
$t_{SU;STA}$	Repeated START set-up time	≥ 14 t_{CLCL}	> 4.7µs [1]
$t_{SU;STO}$	STOP condition set-up time	≥ 14 t_{CLCL}	> 4.0µs [1]
t_{BUF}	Bus free time	≥ 14 t_{CLCL}	> 4.7µs [1]
t_{RD}	SDA rise time	≤ 1µs	– [2]
t_{FD}	SDA fall time	≤ 0.3µs	< 0.3µs [3]

NOTES:
1. At 100 kbit/s. At other bit rates this value is inversely proportional to the bit-rate of 100 kbit/s.
2. Determined by the external bus-line capacitance and the external bus-line pull-resistor, this must be < 1µs.
3. Spikes on the SDA and SCL lines with a duration of less than 3 t_{CLCL} will be filtered out. Maximum capacitance on bus-lines SDA and SCL = 400pF.
4. $t_{CLCL} = 1/f_{OSC}$ = one oscillator clock period at pin XTAL1. For 62ns (42ns) < t_{CLCL} < 285ns (16MHz (24MHz) > f_{OSC} > 3.5MHz) the SI01 interface meets the I²C-bus specification for bit-rates up to 100 kbit/s.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

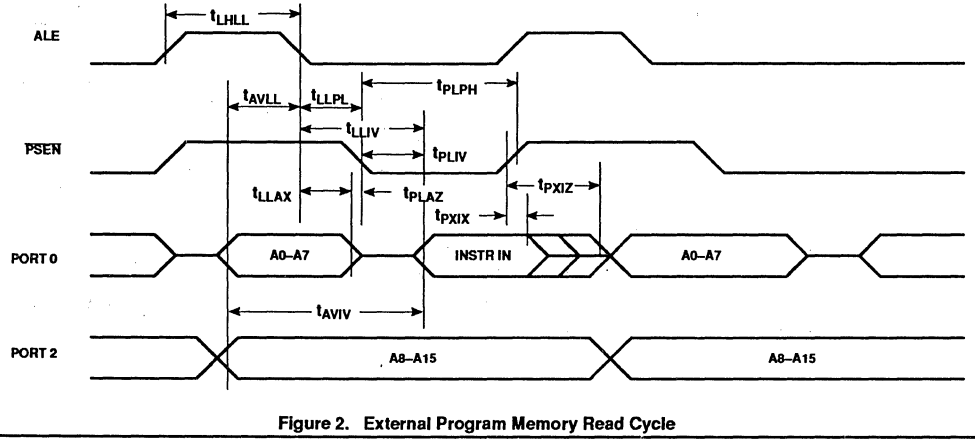


Figure 2. External Program Memory Read Cycle

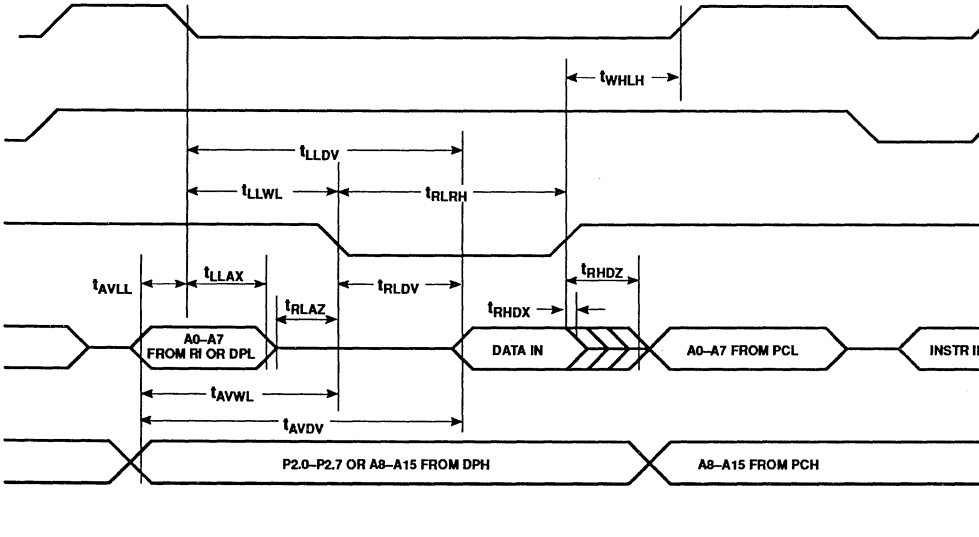


Figure 3. External Data Memory Read Cycle

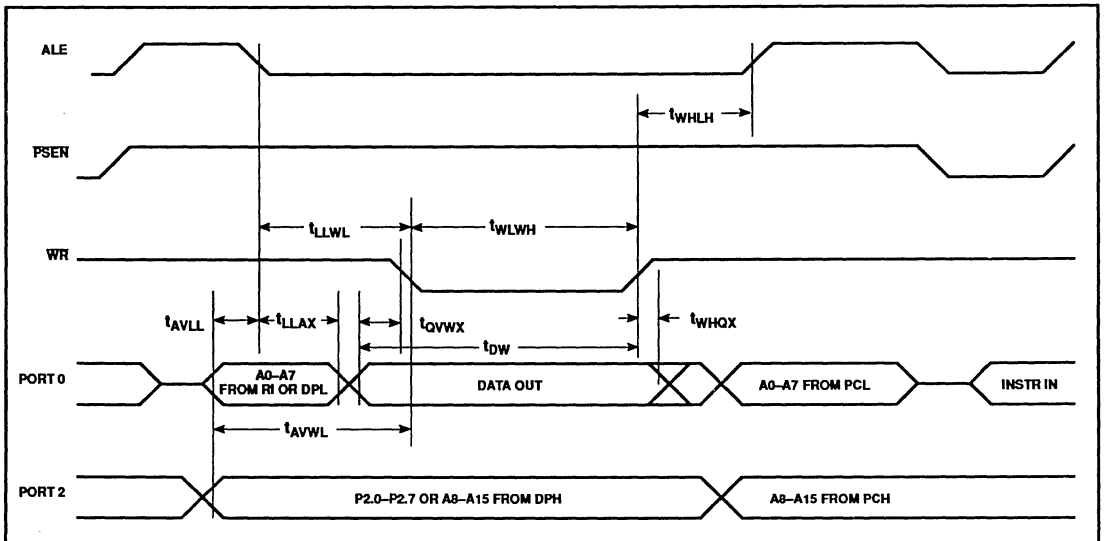


Figure 4. External Data Memory Write Cycle

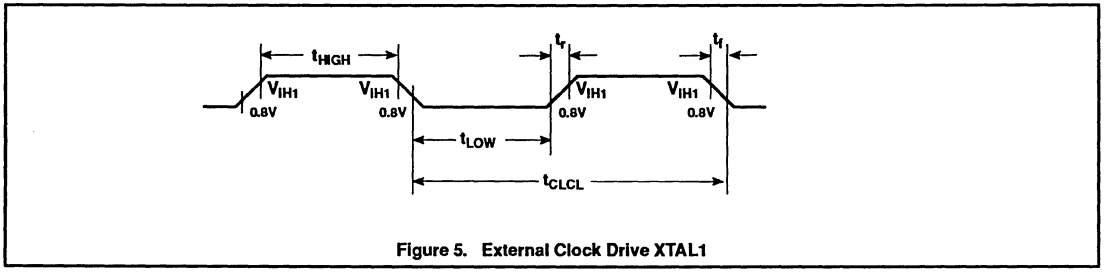


Figure 5. External Clock Drive XTAL1

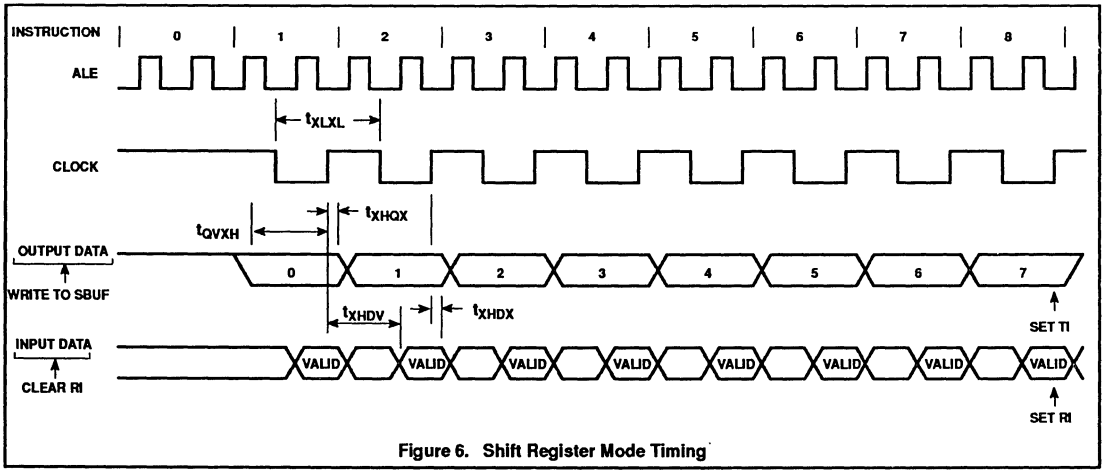
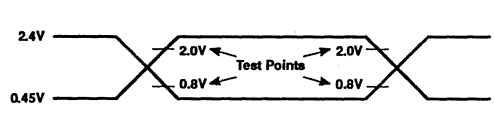
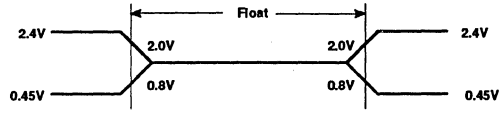


Figure 6. Shift Register Mode Timing



NOTE:
AC inputs during testing are driven at 2.4V for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at 2.0V for a logic '1' and 0.8V for a logic '0'.

Figure 7. AC Testing Input/Output



NOTE:
The float state is defined as the point at which a port 0 pin sinks 3.2mA or sources 400µA at the voltage test levels.

Figure 8. AC Testing Input, Float Waveform

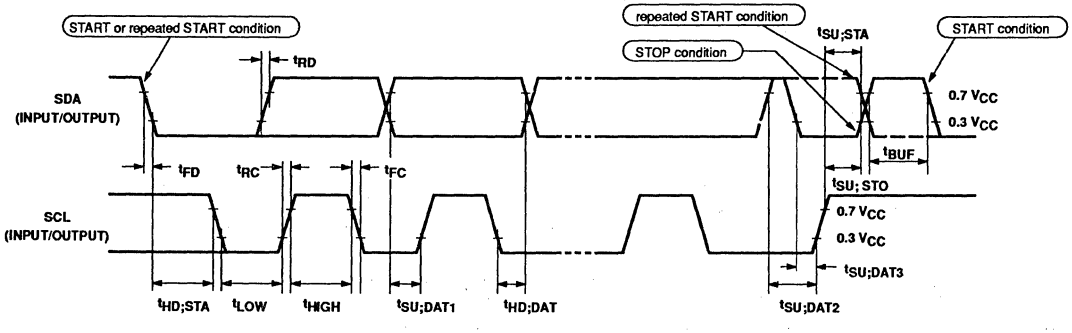
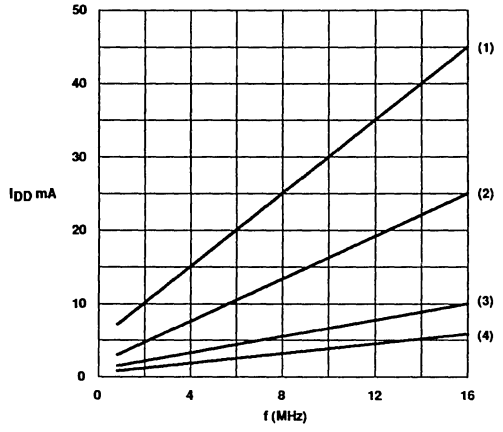


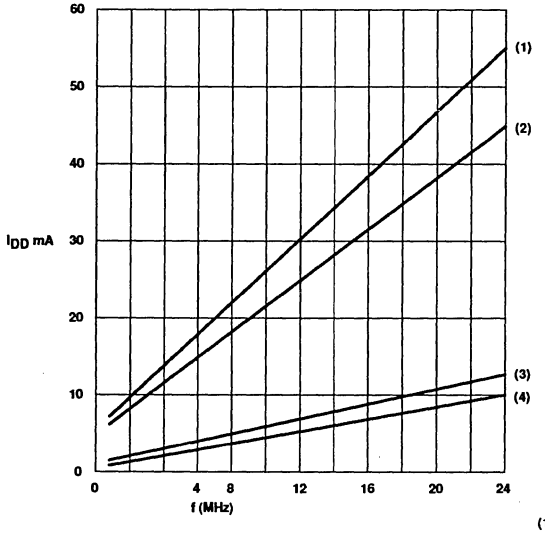
Figure 9. Timing SIO1 (I^2C) Interface



(1) Maximum operating mode; V_{DD} = 6V
(2) Maximum operating mode; V_{DD} = 4V
(3) Maximum idle mode; V_{DD} = 6V
(4) Maximum idle mode; V_{DD} = 4V

NOTE:
These values are valid only within the frequency specifications of the device under test.

Figure 10. 16MHz Version Supply Current (I_{DD}) as a Function of Frequency at XTAL1 (f_{OSC})



(1) Maximum operating mode; V_{DD} = 5.5V
(2) Maximum operating mode; V_{DD} = 4.5V
(3) Maximum idle mode; V_{DD} = 5.5V
(4) Maximum idle mode; V_{DD} = 4.5V

NOTE:
These values are valid only within the frequency specifications of the device under test.

Figure 11. 24MHz Version Supply Current (I_{DD}) as a Function of Frequency at XTAL1 (f_{OSC})

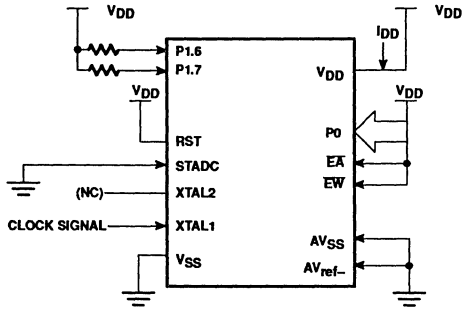


Figure 12. I_{DD} Test Condition, Active Mode
All other pins are disconnected[1]

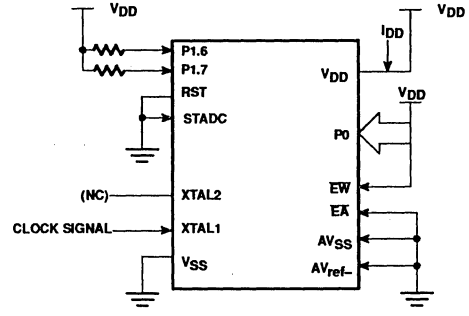


Figure 13. I_{DD} Test Condition, Idle Mode
All other pins are disconnected[2]

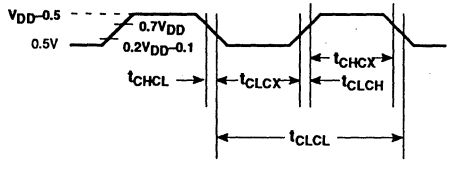


Figure 14. Clock Signal Waveform for I_{DD} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

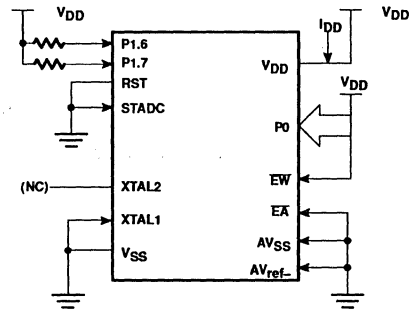


Figure 15. I_{DD} Test Condition, Power Down Mode
All other pins are disconnected. V_{DD} = 2V to 5.5V[3]

NOTES:

1. Active Mode:
   a. The following pins must be forced to V_{DD}: $\overline{EA}$, RST, Port 0, and $\overline{EW}$.
   b. The following pins must be forced to V_{SS}: STADC, AV_{ss}, and AV_{ref-}.
   c. Ports 1.6 and 1.7 should be connected to V_{DD} through resistors of sufficiently high value such that the sink current into these pins cannot exceed the I_{OL1} spec of these pins.
   d. The following pins must be disconnected: XTAL2 and all pins not specified above.
2. Idle Mode:
   a. The following pins must be forced to V_{DD}: Port 0 and $\overline{EW}$.
   b. The following pins must be forced to V_{SS}: RST, STADC, AV_{ss},, AV_{ref-}, and $\overline{EA}$.
   c. Ports 1.6 and 1.7 should be connected to V_{DD} through resistors of sufficiently high value such that the sink current into these pins cannot exceed the I_{OL1} spec of these pins. These pins must not have logic 0 written to them prior to this measurement.
   d. The following pins must be disconnected: XTAL2 and all pins not specified above.
3. Power Down Mode:
   a. The following pins must be forced to V_{DD}: Port 0 and $\overline{EW}$.
   b. The following pins must be forced to V_{SS}: RST, STADC, XTAL1, AV_{ss},, AV_{ref-}, and $\overline{EA}$.
   c. Ports 1.6 and 1.7 should be connected to V_{DD} through resistors of sufficiently high value such that the sink current into these pins cannot exceed the I_{OL1} spec of these pins. These pins must not have logic 0 written to them prior to this measurement.
   d. The following pins must be disconnected: XTAL2 and all pins not specified above.

EPROM CHARACTERISTICS

The 87C552 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C552 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C552 manufactured by Signetics.

Table 3 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 16 and 17. Figure 18 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 16. Note that the 87C552 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 16. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$, and pins of ports 2 and 3 specified in Table 3 are held at the "Program Code Data" levels indicated in Table 3. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 17.

To program the encryption table, repeat the 25-pulse programming sequence for addresses 0 through 1FH, using the "Pgm Encryption Table" levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25-pulse programming sequence using the "Pgm Lock Bit" levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be red is applied to ports 1 and 2 as shown in Figure 18. The other pins are held at the "Verify Code Data" levels indicated in Table 3. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips Components

(031H) = 94H indicates 87C552

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 3, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to the light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345-5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000µW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient. Erasure leaves the array in an all 1s state.

Table 3. EPROM Programming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signagure	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:
1. 0 = Valid low for that pin; 1 = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{DD} = 5V ±10% during programming and verification.

* ALE/$\overline{PROG}$ receives 25 programming pulses whill V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

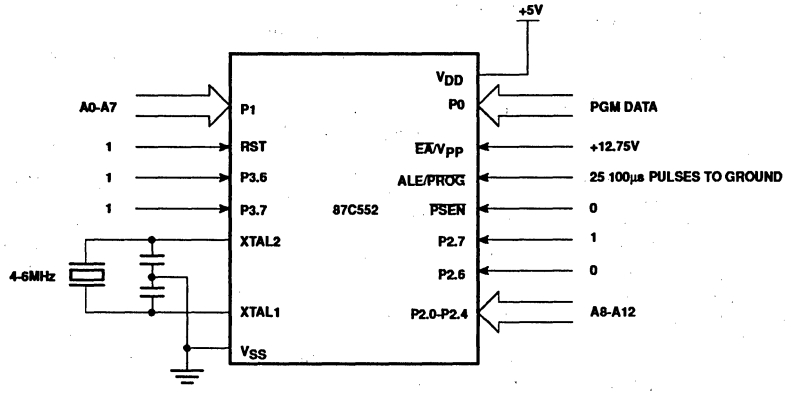


Figure 16. Programming Configuration

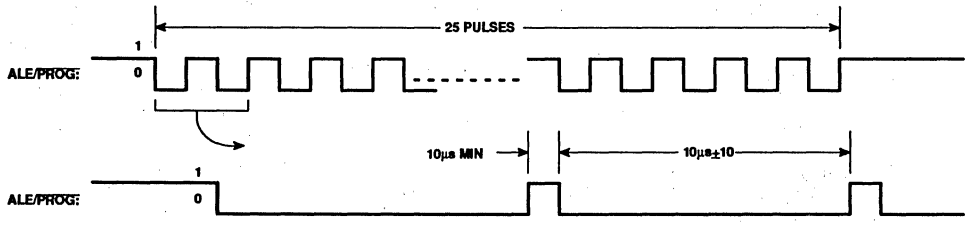


Figure 17. PROG Waveform

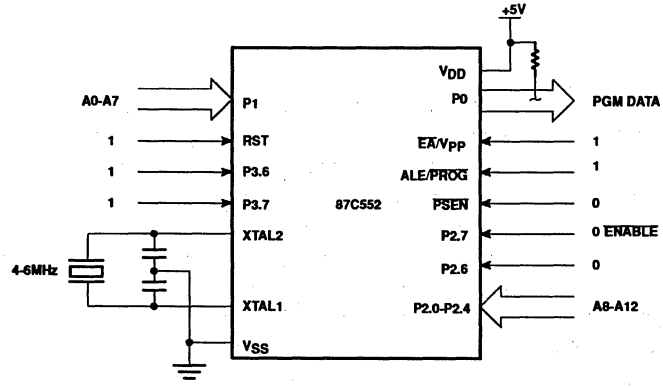


Figure 18. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_{amb} = 21°C to +27°C, V_{DD} = 5V±10%, V_{SS} = 0V (See Figure 19)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

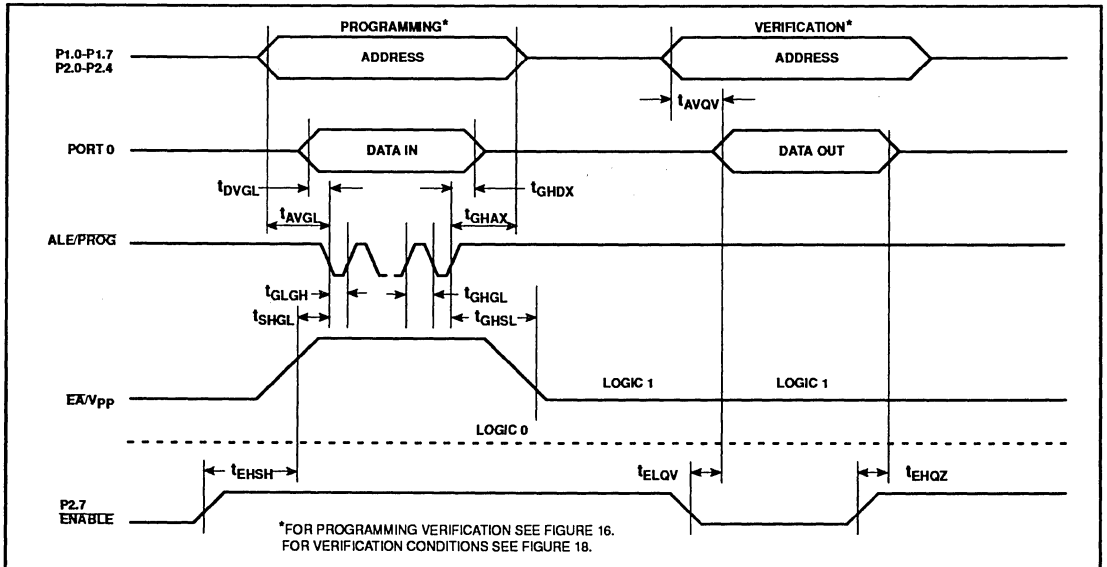


Figure 19. EPROM Programming and Verification

Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specifications defined by Philips.

Single-chip 8-bit microcontroller — 80C562/83C562

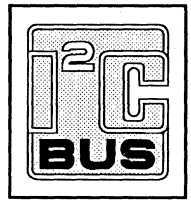
Single-chip 8-bit microcontroller with 8-bit A/D, capture/compare timer, high-speed outputs, PWM

DESCRIPTION

The 80C562/83C562 (hereafter generically referred to as 8XC562) Single-Chip 8-Bit Microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 83C562/83C562 has the same instruction set as the 80C51.

The 8XC562 contains a non-volatile 256 × 8 read-only program memory, a volatile 256 × 8 read/write data memory (83C562) (the 80C562 is ROMless), a volatile 256 × 8 read/write data memory, six 8-bit I/O ports, two 16-bit timer/event counters (identical to the timers of the 80C51), an additional 16-bit timer coupled to capture and compare latches, a 15-source, two-priority-level, nested interrupt structure, an 8-input ADC, two pulse width modulated outputs, standard 80C51 UART, a "watchdog" timer and on-chip oscillator and timing circuits. For systems that require extra capability, the 83C562 can be expanded using standard TTL compatible memories and logic.

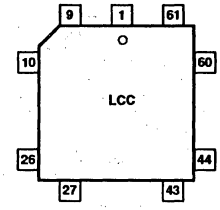
The device also functions as an arithmetic processor having facilities for both binary and BCD arithmetic plus bit-handling capabilities. The instruction set consists of over 100 instructions: 49 one-byte, 45 two-byte and 17 three-byte. With a 12MHz crystal, 58% of the instructions are executed in 1μs and 40% in 2μs. Multiply and divide instructions require 4μs.



FEATURES

- 80C51 instruction set
- 8k × 8 ROM expandable externally to 64k bytes
- 256 × 8 RAM, expandable externally to 64k bytes
- Two standard 16-bit timer/counters
- An additional 16-bit timer/counter coupled to four capture registers and three compare registers
- Capable of producing eight synchronized, timed outputs
- An 8-bit ADC with eight multiplexed analog inputs
- Two 8-bit resolution, pulse width modulated outputs
- Five 8-bit I/O ports plus one 8-bit input port shared with analog inputs
- Full-duplex UART compatible with the standard 80C51
- On-chip watchdog timer
- Three temperature ranges
  - 0 to +70°C
  - –40 to +85°C
  - –40 to +125°C

PIN CONFIGURATION



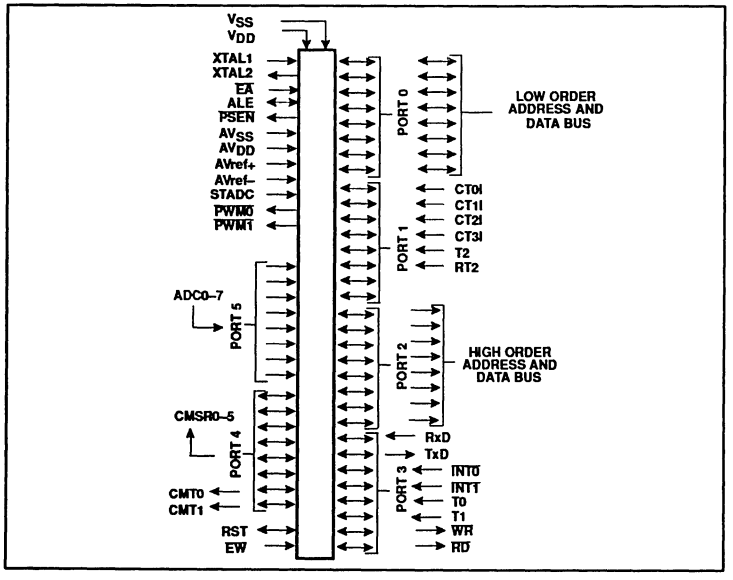
Pin	Function	Pin	Function
1	P5.0/ADC0	35	XTAL1
2	V_{DD}	36	V_{SS}
3	STADC	37	V_{SS}
4	$\overline{PWM0}$	38	NC
5	$\overline{PWM1}$	39	P2.0/A08
6	$\overline{EW}$	40	P2.1/A09
7	P4.0/CMSR0	41	P2.2/A10
8	P4.1/CMRS1	42	P2.3/A11
9	P4.2CMSR2	43	P2.4/A12
10	P4.3/CMSR3	44	P2.5/A13
11	P4.4/CMSR4	45	P2.6/A14
12	P4.5/CMSR5	46	P2.7/A15
13	P4.6/CMT0	47	$\overline{PSEN}$
14	P4.7/CMT1	48	ALE
15	RST	49	$\overline{EA}$
16	P1.0/CT0I	50	P0.7/AD7
17	P1.1/CT1I	51	P0.6/AD6
18	P1.2/CT2I	52	P0.5/AD5
19	P1.3/CT3I	53	P0.4/AD4
20	P1.4/T2	54	P0.3/AD3
21	P1.5/RT2	55	P0.2/AD2
22	P1.6	56	P0.1/AD1
23	P1.7	57	P0.0/AD0
24	P3.0/RxD	58	AVref–
25	P3.1TxD	59	AVref+
26	P3.2/$\overline{INT0}$	60	AV_{SS}
27	P3.3/$\overline{INT1}$	61	AV_{DD}
28	P3.4/T0	62	P5.7/ADC7
29	P3.5/T1	63	P5.6/ADC6
30	P3.6/$\overline{WR}$	64	P5.5/ADC5
31	P3.7/$\overline{RD}$	65	P5.4/ADC4
32	NC	66	P5.3/ADC3
33	NC	67	P5.2/ADC2
34	XTAL2	68	P5.1/ADC1

PART NUMBER SELECTION

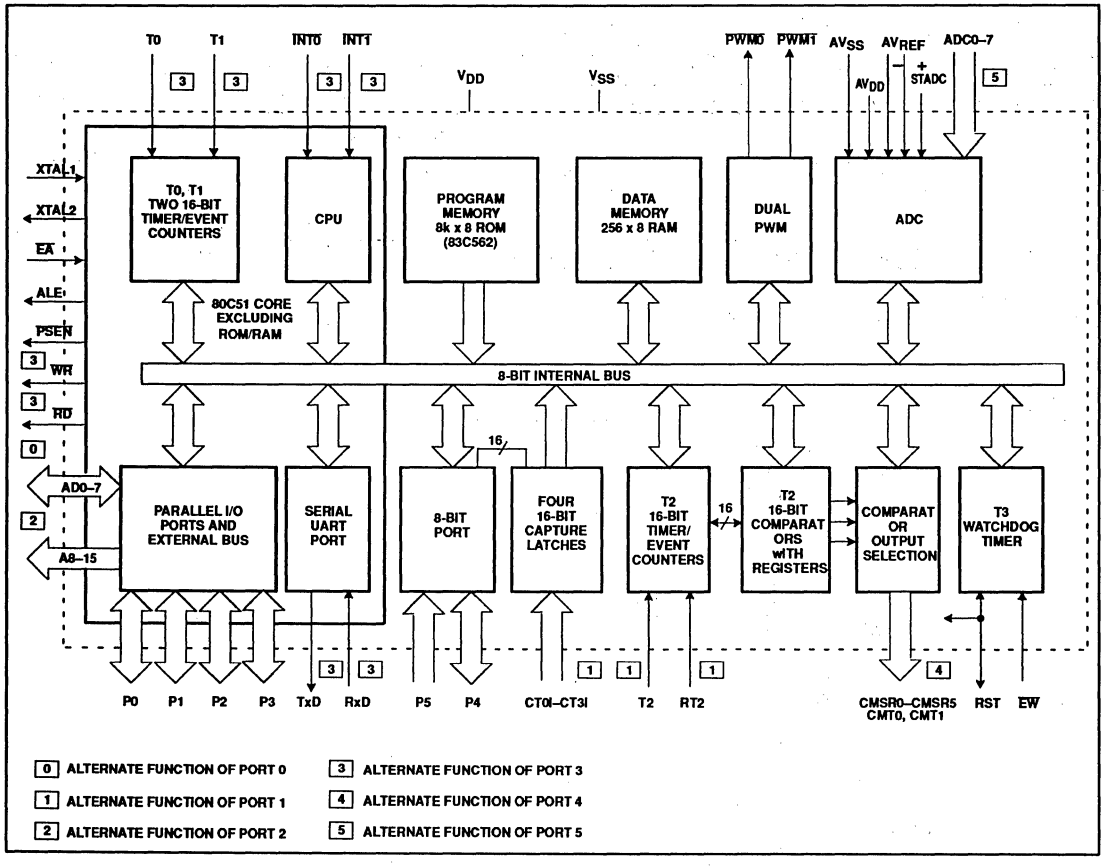
PHILIPS PART ORDER NUMBER PART MARKING		SIGNETICS PART ORDER NUMBER			TEMPERATURE °C	
ROMless	ROM	ROMless	ROM	EPROM	AND PACKAGE	FREQ.
PCB80C562-16WP	PCB83C562-16WP/xxx	S80C562-4A68	S83C562-4A68	S87C552-4A68[2]	0 to +70, plastic PLCC	16MHz
				S87C552-4K68[2]	0 to +70, ceramic CLCC with window	16MHz
PCF80C562-12WP	PCF83C562-12WP/xxx	S80C562-2A68	S83C562-2A68	S87C552-5A68[2]	–40 to +85, plastic PLCC	12MHz
				S87C552-5K68[2]	–40 to +85, ceramic CLCC with window	12MHz
PCA80C562-12WP	PCA83C562-12WP/xxx	S80C562-6A68	S83C562-6A68		–40 to +125, plastic PLCC	12MHz

NOTES:
1. 80C562 and 83C562 frequency range is 1.2MHz–12MHz or 1.2MHz–16MHz.
2. 87C552 frequency range is 3.5MHz–16MHz. For full specification, see the 80C552/83C552/87C552 data sheet.
3. xxx denotes the ROM code number.

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{DD}	2	I	**Digital Power Supply:** +5V power supply pin during normal operation, idle and power-down mode.
STADC	3	I	**Start ADC Operation:** Input starting analog to digital conversion (ADC operation can also be started by software).
$\overline{PWM0}$	4	O	**Pulse Width Modulation:** Output 0.
$\overline{PWM1}$	5	O	**Pulse Width Modulation:** Output 1.
$\overline{EW}$	6	I	**Enable Watchdog Timer:** Enable for T3 watchdog timer and disable power-down mode.
P0.0–P0.7	57–50	I/O	**Port 0:** Port 0 is an 8-bit open-drain bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application it uses strong internal pull-ups when emitting 1s.
P1.0–P1.7	16–23	I/O	**Port 1:** 8-bit I/O port. Alternate functions include:
	16–23	I/O	**(P1.0–P1.7):** Quasi-bidirectional port pins.
	16–19	I/O	**CT0I–CT3I (P1.0–P1.3):** Capture timer input signals for timer T2.
	20	I	**T2 (P1.4):** T2 event input
	21	I	**RT2 (P1.5):** T2 timer reset signal. Rising edge triggered.
P2.0–P2.7	39–46	I/O	**Port 2:** 8-bit quasi-bidirectional I/O port. Alternate function: High-order address byte for external memory (A08–A15).
P3.0–P3.7	24–31	I/O	**Port 3:** 8-bit quasi-bidirectional I/O port. Alternate functions include:
	24		**RxD(P3.0):** Serial input port.
	25		**TxD (P3.1):** Serial output port.
	26		**$\overline{INT0}$ (P3.2):** External interrupt.
	27		**$\overline{INT1}$ (P3.3):** External interrupt.
	28		**T0 (P3.4):** Timer 0 external input.
	29		**T1 (P3.5):** Timer 1 external input.
	30		**$\overline{WR}$ (P3.6):** External data memory write strobe.
	31		**$\overline{RD}$ (P3.7):** External data memory read strobe.
P4.0–P4.7	7–14	I/O	**Port 4:** 8-bit quasi-bidirectional I/O port. Alternate functions include:
	7–12	O	**CMSR0–CMSR5 (P4.0–P4.5):** Timer T2 compare and set/reset outputs on a match with timer T2.
	13, 14	O	**CMT0, CMT1 (P4.6, P4.7):** Timer T2 compare and toggle outputs on a match with timer T2.
P5.0–P5.7	68–62, 1	I	**Port 5:** 8-bit input port. **ADC0–ADC7 (P5.0–P5.7):** Alternate function: Eight input channels to ADC.
RST	15	I/O	**Reset:** Input to reset the 87C552. It also provides a reset pulse as output when timer T3 overflows.
XTAL1	35	I	**Crystal Input 1:** Input to the inverting amplifier that forms the oscillator, and input to the internal clock generator. Receives the external clock signal when an external oscillator is used.
XTAL2	34	O	**Crystal Input 2:** Output of the inverting amplifier that forms the oscillator. Left open–circuit when an external clock is used.
V_{SS}	36, 37	I	**Digital ground.**
$\overline{PSEN}$	47	O	**Program Store Enable:** Active-low read strobe to external program memory.
ALE	48	O	**Address Latch Enable:** Latches the low byte of the address during accesses to external memory. It is activated every six oscillator periods. During an external data memory access, one ALE pulse is skipped. ALE can drive up to eight LS TTL inputs and handles CMOS inputs without an external pull-up.
$\overline{EA}$	49	I	**External Access:** When $\overline{EA}$ is held at TTL level high, the CPU executes out of the internal program ROM provided the program counter is less than 8192. When $\overline{EA}$ is held at TTL low level, the CPU executes out of external program memory. $\overline{EA}$ is not allowed to float.
AV_{REF-}	58	I	**Analog to Digital Conversion Reference Resistor:** Low-end.
AV_{REF+}	59	I	**Analog to Digital Conversion Reference Resistor:** High-end.
AV_{SS}	60	I	**Analog Ground**
AV_{DD}	61	I	**Analog Power Supply**

NOTE:
1. To avoid "latch-up" effect at power-on, the voltage on any pin at any time must not be higher or lower than V_{DD} +0.5V or V_{SS} – 0.5V, respectively.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To ensure a good power-on reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-on, the voltage on V_{DD} and RST must come up at the same time for a proper start-up.

IDLE MODE

In the idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON. Table 1 shows the state of the I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	PSEN	PORT 0	PORT 1	PORT 2	PORT 3	PORT 4	PWM0/ PWM1
Idle	Internal	1	1	Data	Data	Data	Data	Data	High
Idle	External	1	1	Float	Data	Address	Data	Data	High
Power-down	Internal	0	0	Data	Data	Data	Data	Data	High
Power-down	External	0	0	Float	Data	Data	Data	Data	High

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Input, output DC current on any single I/O pin	5.0	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.0	W
Storage temperature range	–65 to +150	°C

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

V_{SS}, AV_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{DD}	Supply voltage				
	PCB8XC562		4.0	6.0	V
	PCF8XC562		4.0	6.0	V
	PCA8XC562		4.5	5.5	V
I_{DD}	Supply current operating:	See notes 1 and 2			
	PCB8XC562	f_{OSC} = 16MHz		45	mA
	PCF8XC562	f_{OSC} = 12MHz		34	mA
	PCA8XC562	f_{OSC} = 12MHz		30	mA
I_{ID}	Idle mode:	See notes 1 and 3			
	PCB8XC562	f_{OSC} = 16MHz		10	mA
	PCF8XC562	f_{OSC} = 12MHz		8	mA
	PCA8XC562	f_{OSC} = 12MHz		7	mA
I_{PD}	Power-down current:	See notes 1 and 4; 2V < V_{PD} < V_{DD} max			
	PCB8XC562			50	µA
	PCF8XC562			50	µA
	PCA8XC562			100	µA
Inputs					
V_{IL}	Input low voltage, except $\overline{EA}$		–0.5	$0.2V_{DD}$–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$		–0.5	$0.2V_{DD}$–0.3	V
V_{IH}	Input high voltage, except XTAL1, RST		$0.2V_{DD}$+0.9	V_{DD}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST		$0.7V_{DD}$	V_{DD}+0.5	V
I_{IL}	Logical 0 input current, ports 1, 2, 3, 4	V_{IN} = 0.45V		–50	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3, 4	See note 5		–650	µA
$\pm I_{IL1}$	Input leakage current, port 0, $\overline{EA}$, STADC, $\overline{EW}$	0.45V < V_I < V_{DD}		10	µA
Outputs					
V_{OL}	Output low voltage, ports 1, 2, 3, 4	I_{OL} = 1.6mA[6]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$, $\overline{PWM0}$, $\overline{PWM1}$	I_{OL} = 3.2mA[6]		0.45	V
V_{OH}	Output high voltage, ports 1, 2, 3, 4	V_{DD} + 5V±10%			
		$-I_{OH}$ = 60µA	2.4		V
		$-I_{OH}$ = 25µA	$0.75V_{DD}$		V
		$-I_{OH}$ = 10µA	$0.9V_{DD}$		V
V_{OH1}	Output high voltage (port 0 in external bus mode, ALE, $\overline{PSEN}$, $\overline{PWM0}$, $\overline{PWM1}$)[7]	V_{DD} + 5V±10%			
		$-I_{OH}$ = 400µA	2.4		V
		$-I_{OH}$ = 150µA	$0.75V_{DD}$		V
		$-I_{OH}$ = 40µA	$0.9V_{DD}$		V
V_{OH2}	Output high voltage (RST)	$-I_{OH}$ = 400µA	2.4		V
		$-I_{OH}$ = 120µA	$0.8V_{DD}$		V
R_{RST}	Internal reset pull-down resistor		50	150	kΩ
C_{IO}	Pin capacitance	Test freq = 1MHz, T_{amb} = 25°C		10	pF

DC ELECTRICAL CHARACTERISTICS (Continued)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
Analog Inputs					
AV_{DD}	Analog supply voltage: PCB8XC562 PCF8XC562 PCA8XC562	 $AV_{DD} = V_{DD} \pm 0.2V$ $AV_{DD} = V_{DD} \pm 0.2V$ $AV_{DD} = V_{DD} \pm 0.2V$	 4.0 4.0 4.5	 6.0 6.0 5.5	 V V V
AI_{DD}	Analog supply current: operating:	Port 5 = 0 to AV_{DD}		1.2	mA
AI_{ID}	Idle mode: PCB8XC562 PCF8XC562 PCA8XC562			 50 50 100	 µA µA µA
AI_{PD}	Power-down mode: PCB8XC562 PCF8XC562 PCA8XC562	$2V < AV_{PD} < AV_{DD}$ max		 50 50 100	 µA µA µA
AV_{IN}	Analog input voltage		$AV_{SS}-0.2$	$AV_{DD}+0.2$	V
AV_{REF}	Reference voltage: AV_{REF-} AV_{REF+}		 $AV_{SS}-0.2$	 $AV_{DD}+0.2$	 V V
R_{REF}	Resistance between AV_{REF+} and AV_{REF-}		5	25	kΩ
C_{IA}	Analog input capacitance			15	pF
t_{ADS}	Sampling time			$6t_{CY}$	µs
t_{ADC}	Conversion time (including sampling time)			$24t_{CY}$	µs
DL_e	Differential non-linearity[8, 9, 10]			±1	LSB
IL_e	Integral non-linearity[8, 11]			±1	LSB
OS_e	Offset error[8, 12]			±1	LSB
G_e	Gain error[8, 13]			0.4	%
M_{CTC}	Channel to channel matching			±1	LSB
C_t	Crosstalk between inputs of port 5[14]	0–100kHz		–60	dB

NOTES:
1. See Figures 8 through 12 for I_{DD} test conditions.
2. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = $\overline{EW}$ = V_{DD}; STADC = V_{SS}.
3. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; XTAL2 not connected; Port 0 = $\overline{EW}$ = V_{DD}; $\overline{EA}$ = RST = STADC = V_{SS}.
4. The power-down current is measured with all output pins disconnected; XTAL2 not connected; Port 0 = $\overline{EW}$ = V_{DD}; $\overline{EA}$ = RST = STADC = XTAL1 = V_{SS}.
5. Pins of ports 1, 2, 3, and 4 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
6. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. I_{OL} can exceed these conditions provided that no single output sinks more than 5mA and no more than two outputs exceed the test conditions.
7. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{DD}$ specification when the address bits are stabilizing.
8. Conditions: $AV_{REF-} = 0V$; $AV_{DD} = 5.0V$, $AV_{REF+} = 5.12V$. ADC is monotonic with no missing codes.
9. The differential non-linearity (DL_e) is the difference between the actual step width and the ideal step width. (See Figure 1.)
10. The ADC is monotonic; there are no missing codes.
11. The integral non-linearity (IL_e) is the peak difference between the center of the steps of the actual and the ideal transfer curve after appropriate adjustment of gain and offset error. (See Figure 1.)
12. The offset error (OS_e) is the absolute difference between the straight line which fits the actual transfer curve (after removing gain error), and a straight line which fits the ideal transfer curve. (See Figure 1.)
13. The gain error (G_e) is the relative difference in percent between the straight line fitting the actual transfer curve (after removing offset error), and the straight line which fits the ideal transfer curve. Gain error is constant at every point on the transfer curve. (See Figure 1.)
14. This should be considered when both analog and digital signals are simultaneously input to port 5.

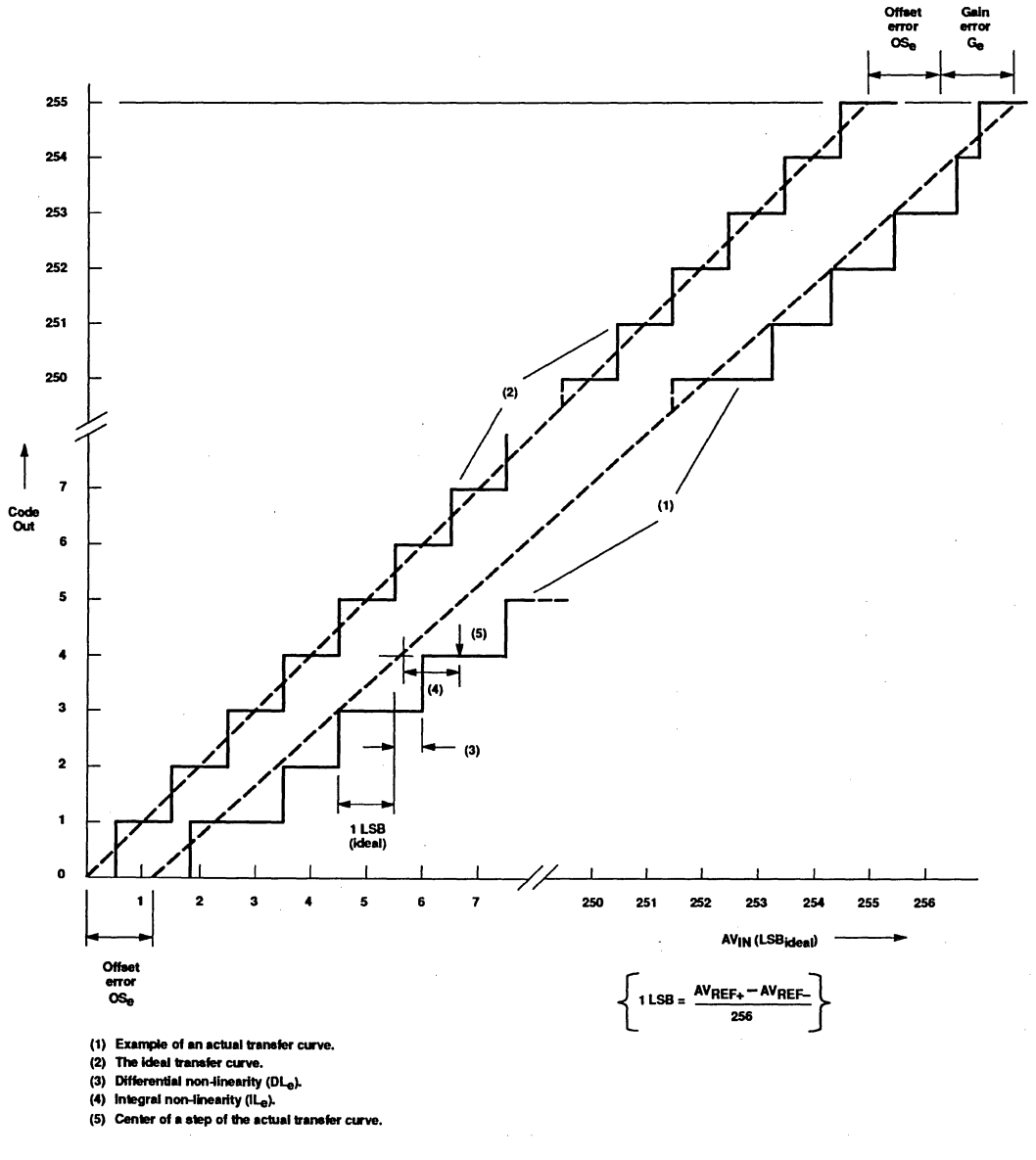


Figure 1. ADC Conversion Characteristic

AC ELECTRICAL CHARACTERISTICS[1, 2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	2	Oscillator frequency			1.2	16	MHz
t_{LHLL}	2	ALE pulse width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	2	Address valid to ALE low	28		$t_{CLCL}-55$		ns
t_{LLAX}	2	Address hold after ALE low	48		$t_{CLCL}-35$		ns
t_{LLIV}	2	ALE low to valid instruction in		234		$4t_{CLCL}-100$	ns
t_{LLPL}	2	ALE low to $\overline{PSEN}$ low	43		$t_{CLCL}-40$		ns
t_{PLPH}	2	$\overline{PSEN}$ pulse width	205		$3t_{CLCL}-45$		ns
t_{PLIV}	2	$\overline{PSEN}$ low to valid instruction in		145		$3t_{CLCL}-105$	ns
t_{PXIX}	2	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	2	Input instruction float after $\overline{PSEN}$		59		$t_{CLCL}-25$	ns
t_{AVIV}	2	Address to valid instruction in		312		$5t_{CLCL}-105$	ns
t_{PLAZ}	2	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{AVLL}	3, 4	Address valid to ALE low	43		$t_{CLCL}-35$		ns
t_{RLRH}	3	$\overline{RD}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	4	$\overline{WR}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	3	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}-165$	ns
t_{RHDX}	3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	3	Data float after $\overline{RD}$		97		$2t_{CLCL}-70$	ns
t_{LLDV}	3	ALE low to valid data in		517		$8t_{CLCL}-150$	ns
t_{AVDV}	3	Address to valid data in		585		$9t_{CLCL}-165$	ns
t_{LLWL}	3, 4	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	3, 4	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}-130$		ns
t_{QVWX}	4	Data valid to $\overline{WR}$ transition	23		$t_{CLCL}-60$		ns
t_{DW}	4	Data before $\overline{WR}$	433		$7t_{CLCL}-150$		ns
t_{WHQX}	4	Data hold after $\overline{WR}$	33		$t_{CLCL}-50$		ns
t_{RLAZ}	3	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	3, 4	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	5	High time[3]	20		20		ns
t_{CLCX}	5	Low time[3]	20		20		ns
t_{CLCH}	5	Rise time[3]		20		20	ns
t_{CHCL}	5	Fall time[3]		20		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. These values are characterized but not 100% production tested.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

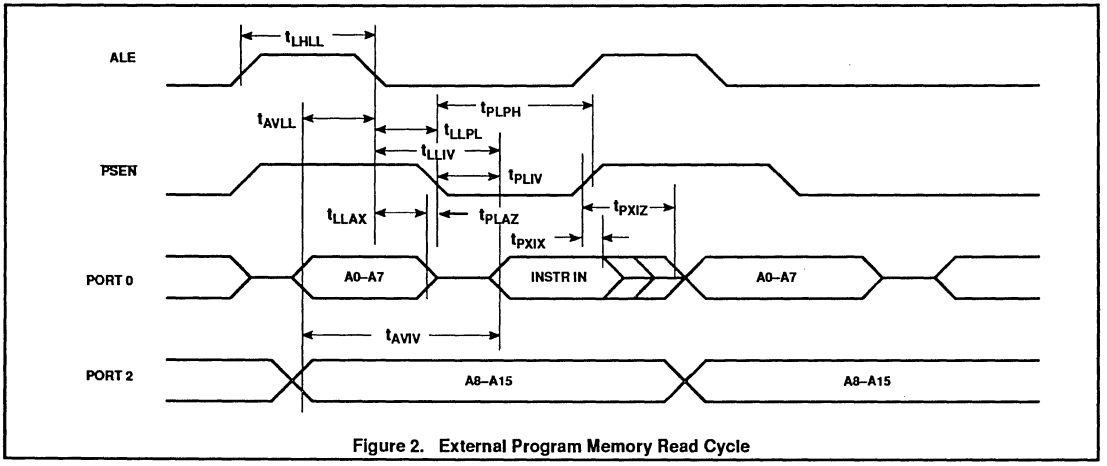


Figure 2. External Program Memory Read Cycle

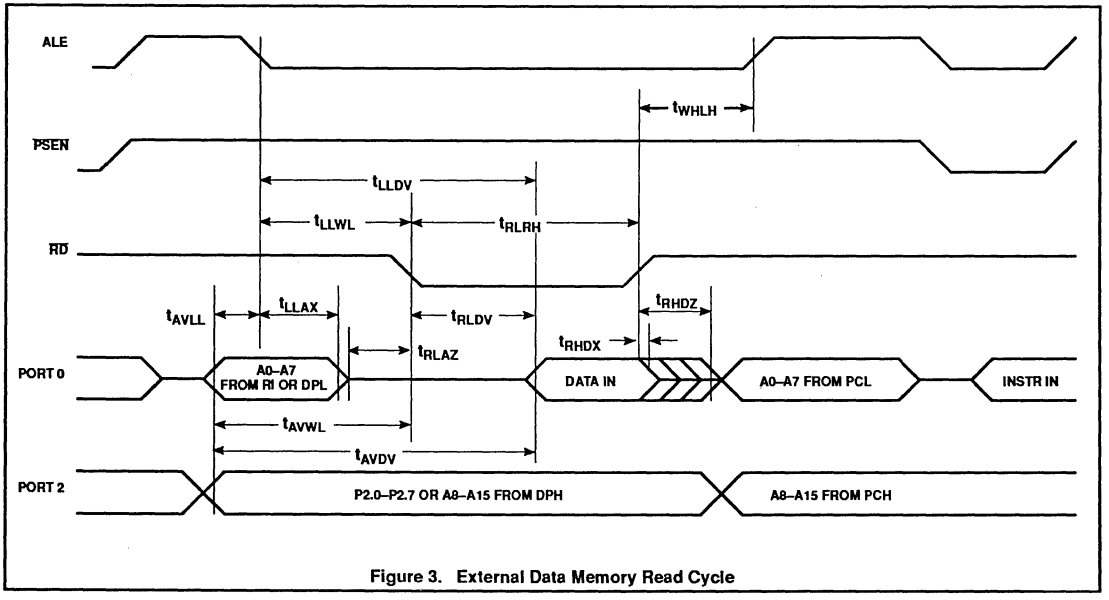


Figure 3. External Data Memory Read Cycle

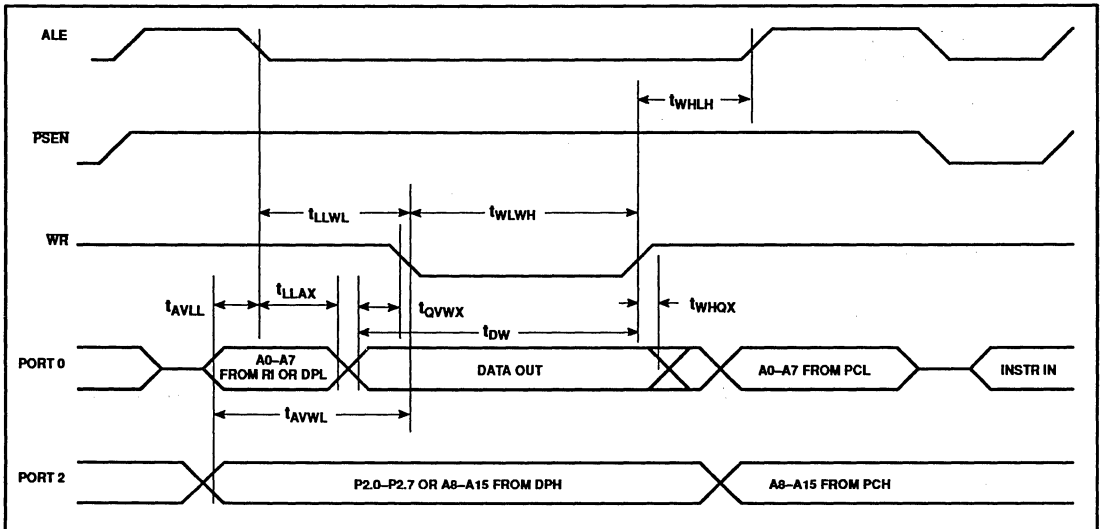


Figure 4. External Data Memory Write Cycle

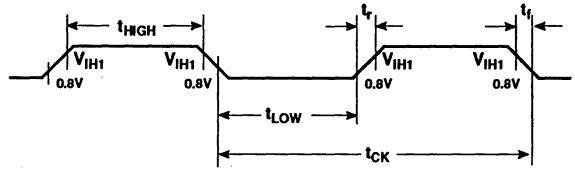
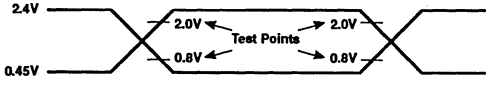
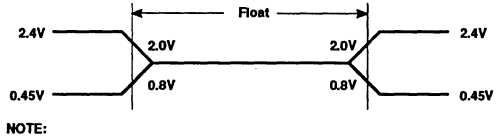


Figure 5. External Clock Drive XTAL1



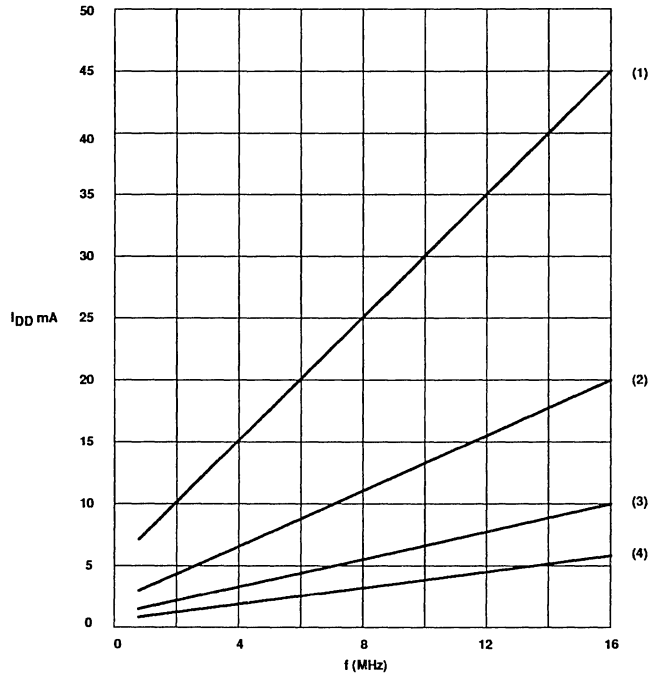
NOTE:
AC inputs during testing are driven at 2.4V for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at 2.0V for a logic '1' and 0.8V for a logic '0'.

Figure 6. AC Testing Input/Output



NOTE:
The float state is defined as the point at which a port 0 pin sinks 3.2mA or sources 400µA at the voltage test levels.

Figure 7. AC Testing Input, Float Waveform



(1) Maximum operating mode; V_{DD} = 6V.
(2) Maximum operating mode; V_{DD} = 4V.
(3) Maximum idle mode; V_{DD} = 6V.
(4) Maximum idle mode; V_{DD} = 4V.

NOTE:
These values are valid only within the frequency specifications of the device under test.

Figure 8. Supply Current (I_{DD}) as a Function of Frequency at XTAL1 (f_{OSC})

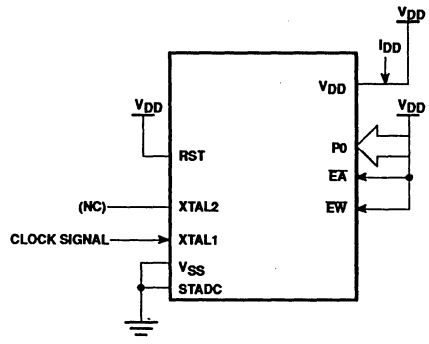


Figure 9. I_{DD} Test Condition, Active Mode
All other pins are disconnected

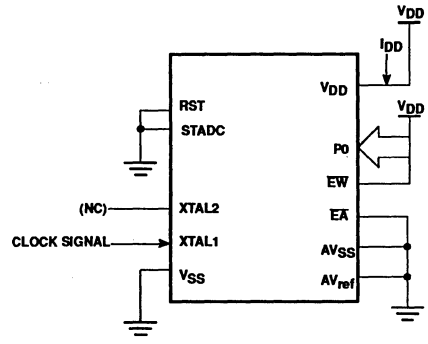


Figure 10. I_{DD} Test Condition, Idle Mode
All other pins are disconnected

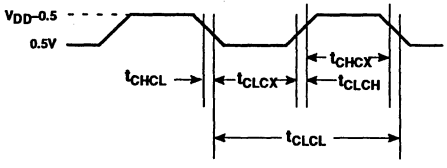


Figure 11. Clock Signal Waveform for I_{DD} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 10ns$

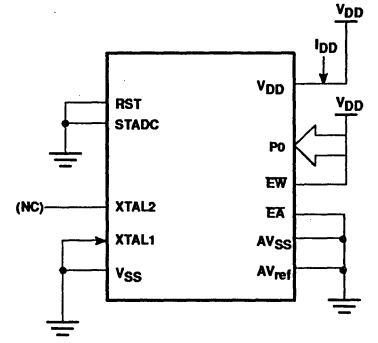


Figure 12. I_{DD} Test Condition, Power Down Mode
All other pins are disconnected. V_{DD} = 2V to 5.5V

Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specifications defined by Philips.

87C575 OVERVIEW

The Signetics 87C575 is a single chip microcontroller derivative of the 80C51. The 87C575 has the same instruction set and core architecture as the industry standard 80C51. The features of the 87C575 include the following:

- 8K bytes EPROM
- 256 bytes RAM
- Two standard 80C51 16-bit Timers
- One 16-bit Timer 2
- Programmable Counter Array
- Watchdog Timer
- Enhanced UART
- Four Analog Comparators
- Low V_{CC} Detect Circuit
- Oscillator Failure Detect Circuit
- Schmitt Trigger Inputs
- Power On Reset Detection Circuit
- Low Active Reset
- Asynchronous Low Port Reset
- Port 2 Selectable Open Drain Output
- Reduced EMI mode
- 40-pin DIP, 44-pin PLCC, 44-pin QFP

Low Active RESET

One of the most notable features on this part is the low active reset. At this time this is the only 80C51 derivative available that has low active reset. This feature makes it easier to interface the 87C575 into an application to accommodate the power-on and low voltage conditions that can occur. The low active reset operates exactly the same as high active reset with the exception that the part is put into the reset mode by applying a low level to the reset pin. For power-on reset it is also necessary to invert the power-on reset circuit; connecting the 8.2K resistor from the reset pin to V_{CC} and the 10µf capacitor from the reset pin to ground.

When reset the port pins on the 87C575 are driven low asynchronously. This is different from all other 80C51 derivatives.

The 87C575 also has Low voltage detection circuitry that will, if enabled, force the part to reset when V_{CC} (on the part) fails below a set level. Low Voltage Reset is enabled by a normal reset. Low Voltage Reset can be disabled by clearing LVRE (bit 5 in the WDCON SFR) then executing a watchdog feed sequency (A5H to WFEED1 followed immediately by 5A to WFEED2). In addition there is a flag (LVF) that is set if a low voltage condition is detected. The LVF flag is set even if the Low Voltage detection circuitry is disabled. Notice that the Low voltage detection circuitry does not drive the RST# pin so the LVF flag is the only way that the microcontroller can determine if it has been reset due to a low voltage condition.

The 87C575 has an on-chip power-on detection circuit that sets the POF (PCON.4) flag on power up or if the V_{CC} level momentarily drops to 0V. This flag can be used to determine if the part is being started from a power-on (cold start) or if a reset has occurred due to another condition (warm start).

Timers

The 87C575 has four on-chip timers.

Timers 0 and 1 are identical in every way to Timers 0 and 1 on the 80C51.

Timer 2 on the 87C575 is identical to the 80C52 Timer 2 (described in detail in the 80C52 overview) with the exception that it is an up or down counter. To configure the Timer to count down the DCEN bit in the T2MOD special function register must be set and a low level must be present on the T2EX pin (P1.1).

The Watchdog timer operation and implementation is the same as that for the 8XC550 (described in the 8XC550 overview) with the exception that the reset values of the WDCON and WDL special function registers have been changed. The changes in these registers cause the watchdog timer to be enabled with a timeout of 98304 × T_{OSC} when the part is reset. The watchdog can be disabled by executing a valid feed sequence and then clearing WDRUN (bit 2 in the WDCON SFR).

Programmable Counter Array (PCA)

The Programmable Counter Array is a special Timer that has five 16-bit capture/compare modules associated with it. Each of the modules can be programmed to operate in one of four modes: rising and/or falling edge capture, software timer, high-speed output, or pulse width modulator. Each module has a pin associated with it in port 1. Module 0 is connected to P1.3(CEX0), module 1 to P1.4(CEX1), etc.

The PCA timer is a common time base for all five modules and can be programmed to run at: 1/12 the oscillator frequency, 1/4 the oscillator frequency, the Timer 0 overflow, or the input on the ECI pin (P1.2). The timer count source is determined from the CPS1 and CPS0 bits in the CMOD SFR as follows:

CPS1	CPS0	PCA Timer Count Source
0	0	1/12 oscillator frequency
0	1	1/4 oscillator frequency
1	0	Timer 0 overflow
1	1	External Input at ECI pin

In the CMOD SFR are three additional bits associated with the PCA. They are CIDL which allows the PCA to stop during idle mode, WDTE which enables or disables the a watchdog function on module 4, and ECF which when set causes an interrupt and the PCA overflow flag CF (in the CCON SFR) to be set when the PCA timer overflows.

The watchdog timer function is implemented in module 4 as implemented in other parts that have a PCA that are available on the market. This industry standard PCA watchdog timer affords little in the way of protection and is very cumbersome to use, so it is recommended that it not be used. If a watchdog timer is required in the target application use the hardware watchdog timer that is implemented on the 87C575 separately from the PCA.

The CCON SFR contains the run control bit for the PCA and the flags for the PCA timer (CF) and each module. To run the PCA the CR bit (CCON.6) must be set by software. The PCA is shut off by clearing this bit. The CF bit (CCON.7) is set when the PCA counter overflows and an interrupt will be generated if the ECF bit in the CMOD register is set, The CF bit can only be cleared by software. Bits 0 through 4 of the CCON register are the flags for the modules (bit 0 for module 0, bit 1 for module 1, etc.) and are set by hardware when either a match or a capture occurs. These flags also can only be cleared by software.

Each module in the PCA has a special function register associated with it. These registers are: CCAPM0 for module 0, CCAPM1 for module 1, etc. The registers contain the bits that control the mode that each module will operate in. The ECCF bit (CCAPMn.0 where n=0, 1, 2, 3, or 4 depending on the module) enables the CCF flag in the CCON SFR to generate an interrupt when a match or compare occurs in the associated module. PWM (CCAPMn.1) enables the pulse width modulation mode. The TOG bit (CCAPMn.2) when set causes the CEX output associated with the module to toggle when there is a match between the PCA counter and the module's capture/compare register. The match bit MAT (CCAPMn.3) when set will cause the CCFn bit in the CCON register to be set when there is a match between the PCA counter and the module's capture/compare register. The next two bits CAPN (CCAPMn.4) and CAPP (CCAPMn.5) determine the edge that a

capture input will be active on. The CAPN bit enables the negative edge, and the CAPP bit enables the positive edge. If both bits are set both edges will be enabled and a capture will occur for either transition. The last bit in the register ECOM (CCAPMn.6) when set enables the comparator function.

There are two additional registers associated with each of the PCA modules. They are CCAPnH and CCAPnL and these are the registers that store the 16-bit count when a capture occurs or a compare should occur. When a module is used in the PWM mode the CCAPnL register is used to control the duty cycle of the output.

PCA Capture Mode
To use one of the PCA modules in the capture mode either one or both of the CCAPM bits CAPN and CAPP for that module must be set. The external CEX input for the module (on port 1) is sampled for a transition. When a valid transition occurs the PCA hardware loads the value of the PCA counter registers (CH and CL) into the module's capture registers (CCAPnL and CCAPnH). If the CCFn bit for the module in the CCON SFR and the ECCFn bit in the CCAPMn SFR are set then an interrupt will be generated.

16-bit Software Timer Mode
The PCA modules can be used as software timers by setting both the ECOM and MAT bits in the modules CCAPMn register. The PCA timer will be compared to the modules capture registers and when a match occurs an interrupt will occur if the CCFn (CCON SFR) and the ECCFn (CCAPMn SFR) bits for the module are both set.

High Speed Output Mode
In this mode the CEX output (on port 1) associated with the PCA module will toggle each time a match occurs between the PCA counter and the module's capture registers. To activate this mode the TOG, MAT, and ECOM bits in the module's CCAPMn SFR must be set.

Pulse Width Modulator Mode
All of the PCA modules can be used as PWM outputs. The frequency of the output depends on the source for the PCA timer. All of the modules will have the same frequency of output because the all share the PCA timer. The duty cycle of each module is independently variable using the module's capture register CCAPLn. When the value of the PCA CL SFR is less than the value in the module's CCAPLn SFR the output will be low, when it is equal to or greater than the output will be high. The PWM and ECOM bits in the module's CCAPMn register must be set to enable the PWM mode.

Enhanced UART

The UART operates in all of the usual modes that are described in the first section of this book for the 80C51. In addition the UART can perform framing error detect by looking for missing stop bits, and automatic address recognition. The 87C575 UART also fully supports multiprocessor communication as does the standard 80C51 UART.

When used for framing error detect the UART looks for missing stop bits in the communication. A missing bit will set the FE bit in the SCON register. The FE bit shares the SCON.7 bit with SM0 and the function of SCON.7 is determine by PCON.6 (SMOD0). If SMOD0 is set then SCON.7 functions as FE. SCON.7 functions as SM0 when SMOD0 is cleared. When used as FE SCON.7 can only be cleared by software.

Analog Comparators

The 87C575 has four analog comparators on-chip. Three of these comparators have a common negative reference. These three comparators have independent positive inputs. The fourth comparator has independent positive and negative inputs.

There are two special function registers associated with the comparators. They are CMP which contains the comparator enables and a bit that can be read by software to determine the state of each comparator's output, and CMPE which controls whether the output from each comparator drives the associated output pin or a capture input associated with one of the PCA modules.

The CMP registers bits 0–3 can be read by software to determine the state of the output of each comparator. To do this the associated comparator must be enabled but the output in port 1 can be disabled. This allows easy polling of the comparator output value without the need to use up a port pin.

The CMPE register allows the comparator to drive the associated PCA module capture input, so that on compare a capture can be generated in the PCA. Bits 0–3 of this register enable the comparator output to drive the associated port 1 output circuitry. Used as a comparator output this circuitry is open drain. To enable the comparator output to drive to port 1, the corresponding port bit must also be set to disable the pulldown. If the comparator is not enabled to drive the port 1 circuitry, the associated port 1 pin can be used for other I/O. This includes when a comparator is enabled to drive the capture input to a PCA module.

Reduced EMI Mode

There are two bits in the AUXR register that can be set to reduce the internal clock drive and disable the ALE output. AO (AUXR.0) when set turns off the ALE output. LO (AUXR.1) when set reduces the drive of the internal clock circuitry. Both bits are cleared on Reset. With LO set the 87C575 will still operate at 12MHz, but will have reduced EMI in the range above 100MHz.

Table 1. 87C575 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
AUXR#	Auxiliary	8EH	–	–	–	–	–	–	LO	AO	xxxxxx00B
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
CCAP0H#	Module 0 Capture High	FAH									xxxxxxxxB
CCAP1H#	Module 1 Capture High	FBH									xxxxxxxxB
CCAP2H#	Module 2 Capture High	FCH									xxxxxxxxB
CCAP3H#	Module 3 Capture High	FDH									xxxxxxxxB
CCAP4H#	Module 4 Capture High	FEH									xxxxxxxxB
CCAP0L#	Module 0 Capture Low	EAH									xxxxxxxxB
CCAP1L#	Module 1 Capture Low	EBH									xxxxxxxxB
CCAP2L#	Module 2 Capture Low	ECH									xxxxxxxxB
CCAP3L#	Module 3 Capture Low	EDH									xxxxxxxxB
CCAP4L#	Module 4 Capture Low	EEH									xxxxxxxxB
CCAPM0#	Module 0 Mode	DAH	–	ECOM	CAPP	CAPN	MAT	TOG	PWM	ECCF	x0000000B
CCAPM1#	Module 1 Mode	DBH	–	ECOM	CAPP	CAPN	MAT	TOG	PWM	ECCF	x0000000B
CCAPM2#	Module 2 Mode	DCH	–	ECOM	CAPP	CAPN	MAT	TOG	PWM	ECCF	x0000000B
CCAPM3#	Module 3 Mode	DDH	–	ECOM	CAPP	CAPN	MAT	TOG	PWM	ECCF	x0000000B
CCAPM4#	Module 4 Mode	DEH	–	ECOM	CAPP	CAPN	MAT	TOG	PWM	ECCF	x0000000B
			DF	DE	DD	DC	DB	DA	D9	D8	
CCON*#	PCA Counter Control	D8H	CF	CR	–	CCF4	CCF3	CCF2	CCF1	CCF0	00x00000B
CH#	PCA Counter High	F9H									00H
CL#	PCA Counter Low	E9H									00H
CMOD#	PCA Counter Mode	D9H	CIDL	WDTE	–	–	–	CPS1	CPS0	ECF	00xxx000B
			EF	EE	ED	EC	EB	EA	E9	E8	
CMP*#	Comparator	E8H	EC3DP	EC2DP	EC1DP	EC0DP	C3RO	C2RO	C1RO	C0RO	00H
CMPE#	Comparator Enable	91H	EC3TDC	EC2TDC	EC1TDC	EC0TDC	EC3OD	EC2OD	EC1OD	EC0OD	00H
DPTR:	Data Pointer (2 bytes)										
DPH	Data Pointer High	83H									00H
DPL	Data Pointer Low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt Enable	A8H	EA	EC	ET2	ES	ET1	EX1	ET0	EX0	00H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt Priority	B8H	–	PPC	PT2	PS	PT1	PX1	PT0	PX0	x0000000B
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	CEX4	CEX3	CEX2	CEX1	CEX0	EXI	T2EX	T2	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	$\overline{RD}$	$\overline{WR}$	T1	T0	$\overline{INT1}$	$\overline{INT0}$	TxD	RxD	FFH

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.

Table 1. 87C575 Special Function Registers (Continued)

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB LSB								RESET VALUE
P2OD#	Port 2 Pullup Disable	A1H									00H
PCON	Power Control	87H	SMOD1	SMOD0	OSF[1]	POF[1]	LVF[1]	GF0	PD	IDL	00xxx000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program Status Word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
RACAP2H#	Timer 2 Capture High	CBH									00H
RACAP2L#	Timer 2 Capture Low	CAH									00H
SADDR#	Slave Address	A9H									00H
SADEN#	Slave Address Mask	B9H									00H
SBUF	Serial Data Buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial Control	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack Pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer Control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IE0	00H
			CF	CE	CD	CC	CB	CA	C9	C8	
T2CON*	Timer 2 Control	C8H	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	00H
T2MOD#	Timer 2 Mode Control	C9H	–	–	–	–	–	–	–	DCEN	xxxxxxx0B
TH0	Timer High 0	8CH									00H
TH1	Timer High 1	8DH									00H
TH2#	Timer High 2	CDH									00H
TL0	Timer Low 0	8AH									00H
TL1	Timer Low 1	8BH									00H
TL2#	Timer Low 2	CCH									00H
TMOD	Timer Mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
			C7	C6	C5	C4	C3	C2	C1	C0	
WDCON*#	Watchdog Timer Control	C0H	PRE2	PRE1	PRE0	LVRE	OFRE	WDRUN	WDTOF	WDMOD	11111101B
WDL#	Watchdog Timer Reload	C1H									00H
WFEED1#	Watchdog Feed 1	C2H									xxH
WFEED2#	Watchdog Feed 2	C3H									xxH

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.
1. Reset value depends on reset source.

CMOS single-chip 8-bit microcontroller — 80C575/83C575/87C575

DESCRIPTION

The Philips 80C575/83C575/87C575 is a high-performance microcontroller fabricated with Philips high-density CMOS technology. The Philips CMOS technology combines the high speed and density characteristics of HMOS with the low power attributes of CMOS. Philips epitaxial substrate minimizes latch-up sensitivity.

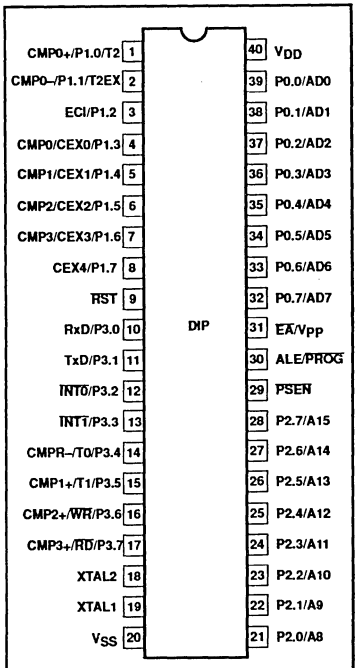
The 8XC575 contains an 8k × 8 ROM (83C575) EPROM (87C575), a 256 × 8 RAM, 32 I/O lines, three 16-bit counter/timers, a Programmable Counter Array (PCA), a seven-source, two-priority level nested interrupt structure, an enhanced UART, four analog comparators, power-fail detect and oscillator fail detect circuits, and on-chip oscillator and clock circuits.

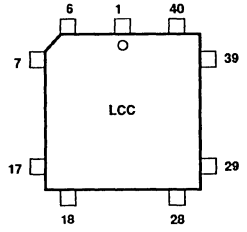
In addition, the 8XC575 has a low active reset, and the port pins are reset to a low level. There is also a fully configurable watchdog timer, and internal power on clear circuit. The part includes idle mode and power-down mode states for reduced power consumption.

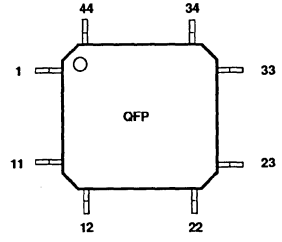
FEATURES

- 80C51 based architecture
  - 8k × 8 ROM (83C575)
  - 8k × 8 EPROM (87C575)
  - ROMless (80C575)
  - 256 × 8 RAM
  - Three 16-bit counter/timers
  - Programmable Counter Array
  - Enhanced UART
  - Boolean processor
  - Oscillator fail detect
  - Low active reset
  - Asynchronous low port reset
  - Schmitt trigger inputs
  - 4 analog comparators
  - Watchdog timer
  - Low V_{CC} detect
- Memory addressing capability
  - 64k ROM and 64k RAM
- Power control modes:
  - Idle mode
  - Power-down mode
- CMOS and TTL compatible
- 4.0 to 16MHz
- Extended temperature ranges
- OTP package available

PIN CONFIGURATIONS





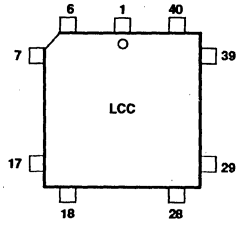


SEE NEXT PAGE FOR LCC AND QFP PIN FUNCTIONS.

PART NUMBER SELECTION

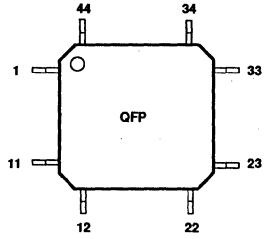
ROMless	ROM	EPROM	TEMPERATURE °C AND PACKAGE	FREQ. (MHz)
P80C575EBPN	P83C575EBPN	P87C575EBPN	0 to +70, plastic DIP	16
P80C575EBAA	P83C575EBAA	P87C575EBAA	0 to +70, plastic LCC	16
		P87C575EBFFA	0 to +70, ceramic DIP	16
		P87C575EBLKA	0 to +70, ceramic LCC	16
P80C575EFPN	P83C575EFPN	P87C575EFPN	–40 to +125, plastic DIP	16
P80C575EFAA	P83C575EFAA	P87C575EFAA	–40 to +125, plastic LCC	16
		P87C575EFFFA	–40 to +125, ceramic DIP	16
		P87C575EFLKA	–40 to +125, ceramic LCC	16
P80C575EBBB	P83C575EBBB	P87C575EBBB	0 to +70, plastic QFP	16

LCC PIN FUNCTIONS



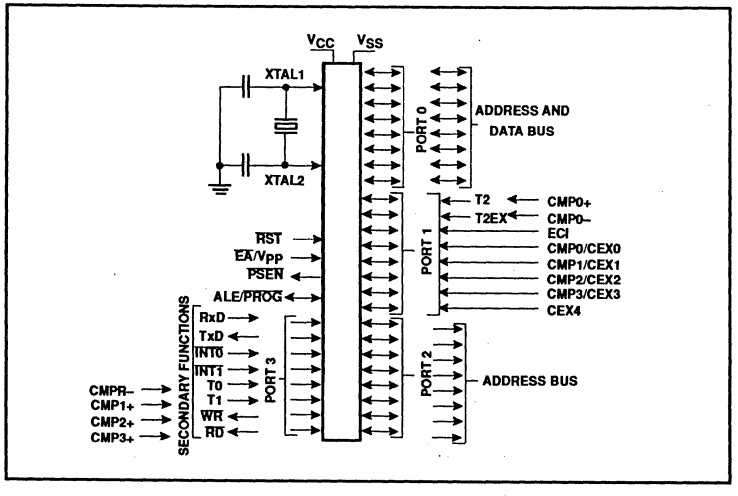
Pin	Function	Pin	Function
1	NC	23	NC
2	T2/P1.0/CMP0+	24	P2.0/A8
3	T2EX/P1.1/CMP0–	25	P2.1/A9
4	P1.2/ECI	26	P2.2/A10
5	P1.3/CMP0/CEX0	27	P2.3/A11
6	P1.4/CMP1/CEX1	28	P2.4/A12
7	P1.5/CMP2/CEX2	29	P2.5/A13
8	P1.6/CMP3/CEX3	30	P2.6/A14
9	P1.7/CEX4	31	P2.7/A15
10	$\overline{RST}$	32	$\overline{PSEN}$
11	RxD/P3.0	33	ALE/$\overline{PROG}$
12	NC	34	NC
13	TxD/P3.1	35	$\overline{EA}$/V_{PP}
14	$\overline{INT0}$/P3.2	36	P0.7/AD7
15	$\overline{INT1}$/P3.3	37	P0.6/AD6
16	T0/P3.4/CMPR–	38	P0.5/AD5
17	T1/P3.5/CMP1+	39	P0.4/AD4
18	$\overline{WR}$/P3.6/CMP2+	40	P0.3/AD3
19	$\overline{RD}$/P3.7/CMP3+	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	V_{SS}	44	V_{CC}

QFP PIN FUNCTIONS

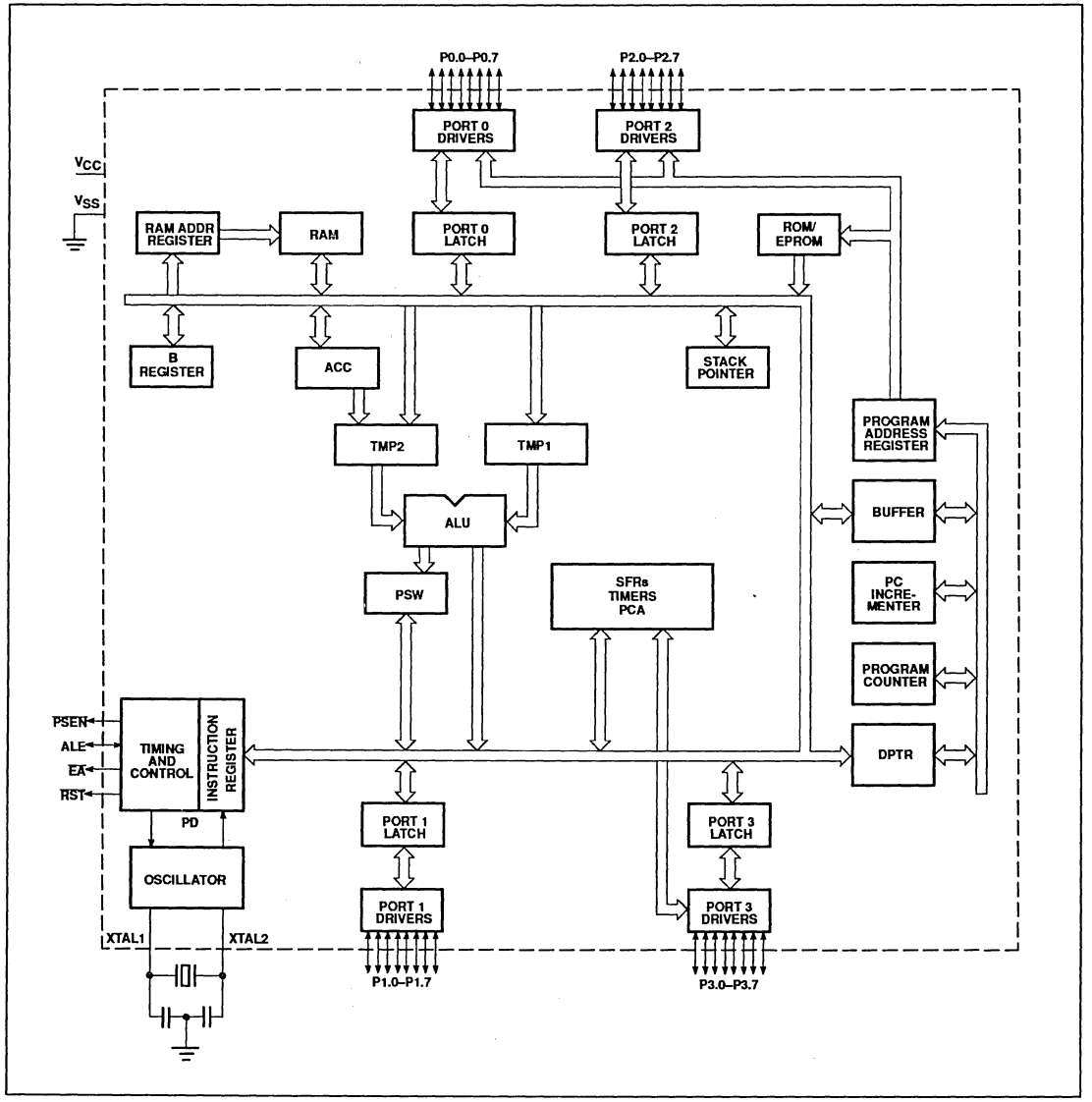


Pin	Function	Pin	Function
1	P1.5/CMP2/CEX2	23	P2.5/A13
2	P1.6/CMP3/CEX3	24	P2.6/A14
3	P1.7/CEX4	25	P2.7/A15
4	$\overline{RST}$	26	$\overline{PSEN}$
5	RxD/P3.0	27	ALE/$\overline{PROG}$
6	NC	28	NC
7	TxD/P3.1	29	$\overline{EA}$/V_{PP}
8	$\overline{INT0}$/P3.2	30	P0.7/AD7
9	$\overline{INT1}$/P3.3	31	P0.6/AD6
10	T0/P3.4/CMPR–	32	P0.5/AD5
11	T1/P3.5/CMP1+	33	P0.4/AD4
12	$\overline{WR}$/P3.6/CMP2+	34	P0.3/AD3
13	$\overline{RD}$/P3.7/CMP3+	35	P0.2/AD2
14	XTAL2	36	P0.1/AD1
15	XTAL1	37	P0.0/AD0
16	V_{SS}	38	V_{CC}
17	NC	39	NC
18	P2.0/A8	40	T2/P1.0/CMP0+
19	P2.1/A9	41	T2EX/P1.1/CMP0–
20	P2.2/A10	42	P1.2/ECI
21	P2.3/A11	43	P1.3/CMP0/CEX0
22	P2.4/A12	44	P1.4/CMP1

LOGIC SYMBOL



BLOCK DIAGRAM



PIN DESCRIPTIONS

MNEMONIC	PIN NUMBER DIP	PIN NUMBER LCC	PIN NUMBER QFP	TYPE	NAME AND FUNCTION
V_{SS}	20	22	16	I	**Ground:** 0V reference.
V_{CC}	40	44	38	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–36	37–30	I/O	**Port 0:** Port 0 is an open-drain bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 also receives code bytes during EPROM programming and outputs code bytes during program verification. External pull-ups are required during program verification. During reset, prot 0 will be asynchronously driven low and will remain low until written to by software. All port 0 pins have Schmitt trigger inputs with 200mV hysteresis. A weak pulldown on port 0 guarantees positive leakage current (see DC Electrical Characteristics: I_{L1}).
P1.0–P1.7	1–8	2–9	40–44 1–3	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port. Port 1 pins have internal pull-ups such that pins that have 1s written to them can be used as inputs but will source current when externally pulled low (see DC Electrical Characteristics: I_{IL}). Port 1 receives the low-order address byte during program memory verification and EPROM programming. During reset, port 1 will be asynchronously driven low and will remain low until written to by software. All port 1 pins have Schmitt trigger inputs with 50mV hysteresis. Port 1 pins also serve alternate functions as follows:
	1	2	40	I/O	**P1.0** **T2** Timer 2 external I/O **CMP0+** Comparator 0 positive input
	2	3	41	I	**P1.1** **T2EX** Timer 2 capture input **CMP0–** Comparator 0 negative input
	3	4	42	I	**P1.2** **ECI** PCA count input
	4	5	43	I/O	**P1.3** **CEX0** PCA module 0 external I/O **CMP0** Comparator 0 output
	5	6	44	I/O	**P1.4** **CEX1** PCA module 1 external I/O **CMP1** Comparator 1 output
	6	7	1	I/O	**P1.5** **CEX2** PCA module 2 external I/O **CMP2** Comparator 2 output
	7	8	2	I/O	**P1.6** **CEX3** PCA module 3 external I/O **CMP3** Comparator 3 output
	8	9	3	I/O	**P1.7** **CEX4** PCA module 4 external I/O
P2.0–P2.7	21–28	24–31	18–25	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them can be used as inputs, but will source current when externally pulled low (see DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during accesses to external program and data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. Port 2 receives the high-order address byte during program verification and EPROM programming. During reset, port 2 will be asynchronously driven low and will remain low until written to by software.
P3.0–P3.7	10–17	11, 13–19	5, 7–13	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins except P3.1 that have 1s written to them can be used as inputs but will source current when externally pulled low (see DC Electrical Characteristics: I_{IL}). P3.1 will be a high impedance pin except while transmitting serial data, in which case the strong pull-up will remain on continuously when outputting a 1 level. The P3.1 output drive level when transmitting can be set to one of two levels by the writing to the P3.1 register bit. During reset all pins (except P3.1) will be asynchronously driven low and will remain low until written to by software. All port 3 pins have Schmitt trigger inputs with 200mV hysteresis, except P3.2 and P3.3, which have 50mV hysteresis. Port 3 pins serve alternate functions as follows:

PIN DESCRIPTIONS (Continued)

MNEMONIC	PIN NUMBER DIP	LCC	QFP	TYPE	NAME AND FUNCTION
					Port 3: (continued)
	10	11	5	I	**P3.0 RxD** Serial receive port
	11	13	7	O	**P3.1 TxD** Serial transmit port
	12	14	8	I	**P3.2 $\overline{INT0}$** External interrupt 0
	13	15	9	I	**P3.3 $\overline{INT1}$** External interrupt 1
	14	16	10	I	**P3.4 T0** Timer/counter 0 input **CMPR–** Common – reference to comparators 1, 2, 3
	15	17	11	I	**P3.5 T1** Timer/counter 1 input **CMP1+** Comparator 1 positive input
	16	18	12	O	**P3.6 $\overline{WR}$** External data memory write strobe **CMP2+** Comparator 2 positive input
	17	19	13	O	**P3.7 $\overline{RD}$** External data memory read strobe **CMP3+** Comparator 3 positive input
$\overline{RST}$	9	10	4	I	**Reset:** A low on this pin asynchronously resets all port pins to a low state except P3.1. The pin must be held low with the oscillator running for 24 oscillator cycles to initialize the internal registers. An internal diffused resistor to V_{CC} permits a power on reset using only an external capacitor to V_{SS}. RST has a Schmitt trigger input stage to provide additional noise immunity with a slow rising input voltage.
ALE/$\overline{PROG}$	30	33	27	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. ALE is switched off if the bit 0 in the AUXR register (8EH) is set. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{PSEN}$	29	32	26	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}/V_{PP}$	31	35	29	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 1FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 1FFFH. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	19	21	15	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	14	O	**Crystal 2:** Output from the inverting oscillator amplifier.

POWER ON CLEAR / POWER ON FLAG

An on-chip Power On Detect Circuit resets the 8XC575 and sets the Power Off Flag (PCON.4) on power up or if V_{CC} drops to zero momentarily. The POF can only be cleared by software. The RST pin is not driven by the power on detect circuit. The POF can be read by software to determine that a power failure has occurred and can also be set by software.

LOW VOLTAGE DETECT

An on-chip Low Voltage Detect circuit sets the Low Voltage Flag (PCON.3) if V_{CC} drops below V_{LOW} (see DC Electrical Characteristics) and resets the 8XC575 if the Low Voltage Reset Enable bit (WDCON.4) is set. If the LVRE is cleared, the reset is disabled but LVF will still be set if V_{CC} is low. The RST pin is not driven by the low voltage detect circuit. The LVF can be read by software to determine that V_{CC} was low. The LVF can be set or cleared by software.

OSCILLATOR FAIL DETECT

An on-chip Oscillator Fail Detect circuit sets the Oscillator Fail Flag (PCON.5) if the oscillator frequency drops below OSCF for one or more cycles (see AC Electrical Characteristics: OSCF) and resets the 8XC575 if the Oscillator Fail Reset Enable bit (WDCON.3) is set. If OFRE is cleared, the reset is disabled but OSF will still be set if the oscillator fails. The RST pin is not driven by the oscillator fail detect circuit. The OSF can be read by software to determine that an oscillator failure has occurred. The OSF can be set or cleared by software.

WATCHDOG TIMER

The watchdog timer is not directly loadable by the user. Instead, the value to be loaded into the main timer is held in an autoload register or is part of the mask ROM programming. In order to cause the main timer to be loaded with the appropriate value, a special sequence of software action must take place. This operation is referred to as feeding the watchdog timer.

To feed the watchdog, two instructions must be sequentially executed successfully. No intervening instruction fetches are allowed, so interrupts should be disabled before feeding the watchdog. The instructions should move A5H to the WFEED1 register and then 5AH to the WFEED2 register. If WFEED1 is correctly loaded and WFEED2 is not correctly loaded, then an immediate underflow will occur.

The watchdog timer subsystem has two modes of operation. Its principal function is a watchdog timer. In this mode it protects the system from incorrect code execution by causing a system reset when the watchdog timer underflows as a result of a failure of software to feed the timer prior to the timer reaching its terminal count. If the user does not employ the watchdog function, the watchdog subsystem can be used as a timer. In this mode, reaching the terminal count sets a flag. In most other respects, the timer mode possesses the characteristics of the watchdog mode. This is done to protect the integrity of the watchdog function.

The watchdog timer subsystem consists of a prescaler and a main counter. The prescaler has 8 selectable taps off the final stages and the output of a selected tap provides the clock to the main counter. The main counter is the section that is loaded as a result of the software feeding the watchdog and it is the section that causes the system reset (watchdog mode) or time-out flag to be set (timer mode) if allowed to reach its terminal count.

Programming the Watchdog Timer

Both the EPROM and ROM devices have a set of SFRs for holding the watchdog autoload values and the control bits. The watchdog time-out flag is present in the watchdog control register and operates the same in all versions. In the EPROM device, the watchdog parameters (autoload value and control) are always taken from the SFRs. In the ROM device, the watchdog parameters can be mask programmed or taken from the SFRs. The selection to take the watchdog parameters from the SFRs or from the mask programmed values is controlled by EA (external access). When EA is high (internal ROM access), the watchdog parameters are taken from the mask programmed values. If the watchdog is mask programmed to the timer mode, then the autoload values and the pre-scaler taps are taken from the SFRs. When EA is low (external access), the watchdog parameters are taken from the SFRs. The user should be able to leave code in his program which initializes the watchdog SFRs even though he has migrated to the mask ROM part. This allows no code changes from EPROM prototyping to ROM coded production parts.

Watchdog Detailed Operation

EPROM Device (and ROMless Operation: EA = 0)

In the ROMless operation (ROM part, EA = 0) and in the EPROM device, the watchdog operates in the following manner.

Whether the watchdog is in the watchdog or timer mode, when external RESET is applied, the following takes place:

- Watchdog mode bit set to watchdog mode.
- Watchdog run control bit set to ON.
- Autoload register set to 00 (min. count).
- Watchdog time-out flag cleared.
- Prescaler is cleared.
- Prescaler tap set to the highest divide.
- Autoload takes place.

The watchdog can be fed even though it is in the timer mode.

Note that the operational concept is for the watchdog mode of operation, when coming out of a hardware reset, the software should load the autoload registers, set the mode to watchdog, and then feed the watchdog (cause an autoload). The watchdog will now be starting at a known point.

If the watchdog is in the watchdog mode and running and happens to underflow at the time the external RESET is applied, the watchdog time-out flag will be cleared.

When the watchdog is in the watchdog mode and the watchdog underflows, the following action takes place:

- Autoload takes place.
- Watchdog time-out flag is set
- Mode bit unchanged.
- Watchdog run bit unchanged.
- Autoload register unchanged.
- Prescaler tap unchanged.
- All other device action same as external reset.

Note that if the watchdog underflows, the program counter will start from 00H as in the case of an external reset. The watchdog time-out flag can be examined to determine if the watchdog has caused the reset condition. The watchdog time-out flag bit can be cleared by software.

When the watchdog is in the timer mode and the timer software underflows, the following action takes place:

- Autoload takes place.
- Watchdog time-out flag is set
- Mode bit unchanged.
- Watchdog run bit unchanged.
- Autoload register unchanged.
- Prescaler tap unchanged.

Mask ROM Device (EA = 1)

In the mask ROM device, the watchdog mode bit (WDMOD) is mask programmed and the bit in the watchdog command register is read only and reflects the mask programmed selection. If the mask programmed mode bit selects the timer mode, then the watchdog run bit (WDRUN) operates as described under EPROM Device. If the mask programmed bit selects the watchdog mode, then the watchdog run bit has no effect on the timer operation.

Watchdog Function

The watchdog consists of a programmable prescaler and the main timer. The prescaler derives its clock from the on-chip oscillator. The prescaler consists of a divide by 12 followed by a 13 stage counter with taps from stage 6 through stage 13. The tap selection is programmable. The watchdog main counter is a down counter clocked (decremented) each time the programmable prescaler underflows. The watchdog generates an underflow signal (and is autoloaded) when the watchdog is at count 0 and the clock to decrement the watchdog occurs. The watchdog is 8 bits long and the autoload value can range from 0 to FFH. (The autoload value of 0 is permissible since the prescaler is cleared upon autoload).

This leads to the following user design equations. Definitions :t_{OSC} is the oscillator period, N is the selected prescaler tap value, W is the main counter autoload value, t_{MIN} is the minimum watchdog time-out value (when the autoload value is 0), t_{MAX} is the maximum

time-out value (when the autoload value is FFH), t_D is the design time-out value.

$t_{MIN} = t_{OSC} \times 12 \times 64$

$t_{MAX} = t_{MIN} \times 128 \times 256$

$t_D = t_{MIN} \times 2^{PRESCALER} \times W$
(where prescaler = 0, 1, 2, 3, 4, 5, 6, or 7)

Note that the design procedure is anticipated to be as follows. A t_{MAX} will be chosen either from equipment or operation considerations and will most likely be the next convenient value higher than t_D. (If the watchdog were inadvertently to start from FFH, an overflow would be guaranteed, barring other anomalies, to occur within t_{MAX}). Then the value for the prescaler would be chosen from:

prescaler = log2 (t_{MAX} / ($t_{OSC} \times 12 \times 256$)) – 6

This then also fixes t_{MIN}. An autoload value would then be chosen from:

$W = t_D / t_{MIN} - 1$

The software must be written so that a feed operation takes place every t_D seconds from the last feed operation. Some tradeoffs may need to be made. It is not advisable to include feed operations in minor loops or in subroutines unless the feed operation is a specific subroutine.

Watchdog Control Register (WDCON) (Bit Addressable) Address C0
The following bits of this register are read only in the ROM part when EA is high: WDMOD, PRE0, PRE1, and PRE2. That is, the register will reflect the mask programmed values. In the ROM part with EA high, these bits are taken from mask coded bits and are not readable by the program. WDRUN is read only in the ROM part when EA is high and WDMOD is in the watchdog mode. When WDMOD is in the timer mode, WDRUN functions normally.

The parameters written into WDMOD, PRE0, PRE1, and PRE2 by the program are not applied directly to the watchdog timer subsystem. The watchdog timer subsystem is directly controlled by a second register which stores these bits. The transfer of these bits from the user register (WDMOD) to the second control register takes place when the watchdog is fed. This prevents random code execution from directly foiling the watchdog function. This does not affect the operation where these bits are taken from mask coded values.

The reset values of the WDCON and WDL registers will be such that the timer resets to the watchdog mode with a timeout period of $12 \times 64 \times 128 \times t_{OSC}$. The watchdog timer will not generate an interrupt. Additional bits in WDCON are used to disable reset generation by the oscillator fail and low voltage detect circuits. WDCON can be written by software only by executing a valid watchdog feed sequence.

WDCON Register Bit Definitions

WDCON.7	PRE2	Prescaler Select 2, reset to 1
WDCON.6	PRE1	Prescaler Select 1, reset to 1
WDCON.5	PRE0	Prescaler Select 0, reset to 1
WDCON.4	LVRE	Low Voltage Reset Enable, reset to 1 (enabled)
WDCON.3	OFRE	Oscillator Fail Reset Enable, reset to 1 (enabled)
WDCON.2	WDRUN	Watchdog Run, reset to 1 (enabled)
WDCON.1	WDTOF	Watchdog Timeout Flag, reset to 0
WDCON.0	WDMOD	Watchdog Mode, reset to 1 (watchdog mode)

INTERNAL RESET

Internal resets generated by the power on, low voltage, and oscillator fail detect circuits are self timed to guarantee proper initialization of the 8XC575. Reset will be held approximately 24 oscillator periods after normal conditions are detected by all enabled detect circuits. Internal resets do not drive RST but will cause missing pulses on ALE.

ANALOG COMPARATORS

Four analog comparators are provided on chip. three comparators have a common negative reference CMPR– and independent positive inputs CMP1+, CMP2+, CMP3+ on port 3. The fourth comparator has independent positive and negative inputs CMP0+ and CMP0– on port 1. The CMP register contains an output and enable bit for each comparator. The CMP register is bit addressable and is located at SFR address E8H.

Pullups at the comparator input pins will be disabled by hardware when the comparator is enabled. In addition, to make inputs high impedance, the corresponding port SFR bits must be set by software to disable the pulldowns.

CMP Register Bit Definitions

CMP.7	enable comparator 3, disable pullups at P3.4, P3.7
CMP.6	enable comparator 2, disable pullups at P3.4, P3.6
CMP.5	enable comparator 1, disable pullups at P3.4, P3.5
CMP.4	enable comparator 0, disable pullups at P1.0, P1.1
CMP.3	comparator 3 output (readonly)
CMP.2	comparator 2 output (readonly)
CMP.1	comparator 1 output (readonly)
CMP.0	comparator 0 output (readonly)

All comparators are disabled automatically in power down mode, in idle mode unused comparators should be disabled by software to save power. A comparator can generate an interrupt that will terminate idle mode when used to drive a PCA capture input.

The CMPE register contains bits to enable each comparator to drive external output pins or internal PCA capture inputs. Pullups at the output pins are disabled by hardware when the external comparator output is enabled. The comparator output is wire-ORed with the corresponding port SFR bit, so the SFR bit must also be set by software to enable the output.

CMPE Register Bit Definitions

CMPE.7	enables comparator 3 to drive CEX3
CMPE.6	enables comparator 2 to drive CEX2
CMPE.5	enables comparator 1 to drive CEX1
CMPE.4	enables comparator 0 to drive CEX0
CMPE.3	enables comparator 3 output on P1.6 (open drain)
CMPE.2	enables comparator 2 output on P1.5 (open drain)
CMPE.1	enables comparator 1 output on P1.4 (open drain)
CMPE.0	enables comparator 0 output on P1.3 (open drain)

When 1s are written to CMPE bits 7–4, the comparator outputs will drive the corresponding capture input. When 1s are written to CMPE bits 3–0 the comparator output will also drive the corresponding port 1 pin. If the comparator s enabled to drive the capture input but not the port pin, then the port pin can be used for general purpose I/O. When a comparator output is enabled, pullups at the output pin are disabled and the output becomes open drain.

Interrupt Enable (IE) Register

EA	IE.7	enable all interrupts
EC	IE.6	enable PCA interrupt
ET2	IE.5	enable Timer 2 interrupt
ES	IE.4	enable Serial I/O interrupt
ET1	IE.3	enable Timer 1 interrupt
EX1	IE.2	enable External interrupt 1
ET0	IE.1	enable Timer 0 interrupt
EX0	IE.0	enable External interrupt 0

Interrupt Priority (IP) Register

	IP.7	reserved
PPC	IP.6	PCA interrupt priority
PT2	IP.5	Timer 2 interrupt priority
PS	IP.4	Serial I/O interrupt priority
PT1	IP.3	Timer 1 interrupt priority
PX1	IP.2	External interrupt 1 priority
PT0	IP.1	Timer 0 interrupt priority
PX0	IP.0	External interrupt 0 priority

Priority	Source	Flag	Vector
1	INT0	IE0	03H highest priority
2	Timer 0	TF0	0BH
3	INT1	IE1	13H
4	Timer 1	TF1	1BH
5	PCA	CF,CCFn	33H
6	Serial I/O	RI,TI	23H
7	Timer 2	TF2/EXF2	2BH lowest priority

Power Control (PCON) Register

SMOD1	PCON.7	double baud rate bit
SMOD0	PCON.6	SCON.7 access control
OSF	PCON.5	oscillator fail flag
POF	PCON.4	power off flag
LVF	PCON.3	low voltage flag
GF0	PCON.2	general purpose flag
PD	PCON.1	power down mode bit
IDL	PCON.0	idle mode bit

Auxilliary Register Bit Definitions (AUXR = 8EH)

AO	AUXR.0	ALE Off, when set turns off ALE
LO	AUXR.1	Low Speed, reduces internal clock drive

Port 2 Pullup Disable Register (P2OD = 0A1H)

Port 2 pullups can be disabled by writing ones to P2OD. Each bit in P2OD controls the corresponding bit in P2. P2OD resets to all zeros enabling Port 2 pullups. Writing one to a P2OD bit disables pullups at the corresponding port 2 bit making the output open drain.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the Logic Symbol, page 462.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. The control bits for the reduced power modes are in the special function register PCON. Power-down mode can be terminated with either a hardware reset or external interrupt. With an external interrupt $\overline{INT0}$ or $\overline{INT1}$ must be enabled and configured as level sensitive. Holding the pin low restarts to oscillator and bringing the pin back high completes the exit.

DESIGN CONSIDERATIONS

At power-on, the voltage on V_{CC} must come up with $\overline{RST}$ low for a proper start-up.

Table 1 shows the state of I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

ROM CODE SUBMISSION

When submitting ROM code for the 83C575, the following must be specified:

1. 8k byte user ROM data
2. 32 byte ROM encryption key
3. ROM security bits
4. The watchdog timer parameters.

ADDRESS	CONTENT	BIT(S)	COMMENT
0000H to 1FFFH	DATA	7:0	User ROM Data
2000H to 201FH	KEY	7:0	ROM Encryption Key FFH = no encryption
2020H	SEC	0	ROM Security Bit 1
2020H	SEC	1	ROM Security Bit 2 0 = enable security 1 = disable security
2030H	WMOD	0	Watchdog mode bit; 00H = timer mode 01H = watchdog mode
2031H	PRE2:0	2:0	Watchdog prescaler selection; 00H = divide by 12 × 64 07H = divide by 12 × 64 × 128 (see specification)
2032H	WD	7:0	Watchdog autoload value (see specification)

Security Bit 1: When programmed, this bit has two effects on masked ROM parts:

1. External MOVC is disabled, and
2. EA# is latched on Reset.

Security Bit 2: When programmed, this bit inhibits Verify User ROM.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Operating temperature under bias	–55 to +125	°C
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ pin to V_{SS}	0 to +13.0	V
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Maximum i_{OL} per I/O pin	15	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.5	W

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_{amb} = –55°C to +125°C, V_{CC} = 5V ±20%, V_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			MIN	TYP[1]	MAX	
V_{IL}	Input low voltage (Ports 0, 2, 3, except 3.2, 3.3)		–0.5		$0.5V_{CC}$–0.6	V
V_{IL1}	Input low voltage (Ports 1, 3.2, 3.3)		–0.5		$0.6V_{CC}$–0.2	V
V_{IL2}	Input low voltage ($\overline{EA}$)		0		$0.2V_{CC}$–0.45	V
V_{IL3}	Input low voltage (XTAL1)		–0.5		$0.2V_{CC}$–0.1	V
V_{IL4}	Input low voltage (RST)		–0.5		$0.4V_{CC}$	V
V_{IH}	Input high voltage (Ports 0, 2, 3, except 3.2, 3.3)		$0.5V_{CC}$+0.8		V_{CC}+0.5	V
V_{IH1}	Input high voltage (Ports 1, 3.2, 3.3)		$0.7V_{CC}$+0.5		V_{CC}+0.5	V
V_{IH2}	Input high voltage ($\overline{EA}$)		$0.2V_{CC}$+0.9		V_{CC}+0.5	V
V_{IH3}	Input high voltage (XTAL1)		$0.7V_{CC}$		V_{CC}+0.5	V
V_{IH4}	Input high voltage (RST)		V_{CC}–1.0		V_{CC}+0.5	V
HYS	Hysteresis (Ports 0, 2, 3, except 3.2, 3.3)		200			mV
HYS1	Hysteresis (Ports 1, 3.2, 3.3)		50			mV
V_{OL}	Output voltage low (Ports 1, 2, 3, except 3.1)	I_{OL} = 1.6mA			0.45	V
V_{OL1}	Output voltage low (Ports 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2mA			0.45	V
V_{OL2}	Output voltage low P3.1 with bit cleared P3.1 with bit set	 I_{OL} = 10.0mA I_{OL} = 3.2mA			 0.50 0.50	 V V
V_{OH}	Output voltage high (Ports 1, 2, 3, except P3.1)	I_{OH} = –100µA I_{OH} = –30µA I_{OH} = –10µA	V_{CC}–1.5 V_{CC}–0.7 V_{CC}–0.3			V V V
V_{OH1}	Output voltage high (Port 0 in external bus mode, ALE, $\overline{PSEN}$)	I_{OH} = –3.2mA I_{OH} = –200µA	V_{CC}–0.7 V_{CC}–0.3			V V
V_{OH1}	Output voltage high P3.1 with bit cleared P3.1 with bit set	 I_{OH} = –10.0mA I_{OH} = –3.2mA	 V_{CC}–1.5 V_{CC}–1.5			 V V
V_{IO}	Offset voltage comparator inputs		–35		+35	mV
V_{CR}	Common mode range comparator inputs		0		V_{CC}	V
I_{IL}	Logical 0 input current (Ports 1, 2, 3, except 3.1)	V_{IN} = 0.45V			–75	µA
I_{TL}	Logical 1-to-0 transition current (Ports 2, 3, except 3.1, 3.2, 3.3) [4]	V_{IN} = $0.5V_{CC}$+0.8			–600	µA
I_{TL1}	Logical 1-to-0 transition current (Ports 1, 3.2, 3.3)	V_{IN} = $0.7V_{CC}$+0.5			–450	µA
I_{L1}	Input leakage current (Port 0)	0.45 < V_{IN} < V_{CC}	2		40	µA
I_{L2}	Input leakage current ($\overline{EA}$, P3.1)	0.45 < V_{IN} < V_{CC}	–10		+10	µA
I_{LC}	Input leakage current comparator inputs	0 < V_{IN} < V_{CC}	–10		+10	µA
I_{CC}	Power supply current:[7] Active mode @ 16MHz[5] Idle mode @ 16MHz Power-down mode	See note 6		 15 5 5	 30 7.5 75	 mA mA µA
R_{RST}	Internal reset pull-up resistor	V_{IN} = 0V	50		200	kΩ
V_{LOW}	Low V_{CC} detect voltage		4.0		4.45	V
C_{IO}	Pin capacitance	f = 1MHz			10	pF

NOTES: (SEE NEXT PAGE)

NOTES TO THE DC ELECTRICAL CHARACTERISTICS TABLE:

1. Typical ratings are not guaranteed. The values listed are at room temperature, 5V.
2. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. I_{OL} can exceed these conditions provided that no single output sinks more than 5mA and no more than two outputs exceed the test conditions.
3. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
4. Pins of ports 1, 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
5. I_{CC}MAX at other frequencies can be determined from Figure 8.
6. See Figures 9 through 12 for I_{CC} test conditions.
7. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
8. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 10mA
   Maximum I_{OL} per 8-bit port: 26mA
   Maximum total I_{OL} for all outputs: 71mA
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

AC ELECTRICAL CHARACTERISTICS

T_{amb} = –55°C to +125°C, V_{CC} = 5V ±20%, V_{SS} = 0V[1,2]

SYMBOL	FIGURE	PARAMETER	VARIABLE CLOCK MIN	VARIABLE CLOCK MAX	UNIT
$1/t_{CLCL}$	1	Oscillator frequency: **Speed Versions** 8XC575 E	4	16	MHz
OSCF		Oscillator fail detect frequency	1	3.9	MHz
TR		Comparator response time		10	µs
t_{LHLL}	1	ALE pulse width	$2t_{CLCL}$–40		ns
t_{AVLL}	1	Address valid to ALE low	t_{CLCL}–13		ns
t_{LLAX}	1	Address hold after ALE low	t_{CLCL}–20		ns
t_{LLIV}	1	ALE low to valid instruction in		$4t_{CLCL}$–65	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	t_{CLCL}–13		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	$3t_{CLCL}$–20		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		$3t_{CLCL}$–45	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		t_{CLCL}–10	ns
t_{AVIV}	1	Address to valid instruction in		$5t_{CLCL}$–55	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10	ns
Data Memory					
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	$6t_{CLCL}$–100		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	$6t_{CLCL}$–100		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		$5t_{CLCL}$–90	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		$2t_{CLCL}$–28	ns
t_{LLDV}	2, 3	ALE low to valid data in		$8t_{CLCL}$–150	ns
t_{AVDV}	2, 3	Address to valid data in		$9t_{CLCL}$–165	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	$3t_{CLCL}$–50	$3t_{CLCL}$+50	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	$4t_{CLCL}$–130		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	t_{CLCL}–40		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	t_{CLCL}–40		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	t_{CLCL}–40	t_{CLCL}+25	ns
External Clock					
t_{CHCX}	5	High time	12		ns
t_{CLCX}	5	Low time	12		ns
t_{CLCH}	5	Rise time		20	ns
t_{CHCL}	5	Fall time		20	ns
Shift Register					
t_{XLXL}	4	Serial port clock cycle time	$12t_{CLCL}$		ns
t_{QVXH}	4	Output data setup to clock rising edge	$10t_{CLCL}$–133		ns
t_{XHQX}	4	Output data hold after clock rising edge	$2t_{CLCL}$–80		ns
t_{XHDX}	4	Input data hold after clock rising edge	0		ns
t_{XHDV}	4	Clock rising edge to input data valid		$10t_{CLCL}$–133	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. Interfacing the 80C32/52 to devices with float times up to 45ns is permitted. This limited bus contention will not cause damage to Port 0 drivers.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

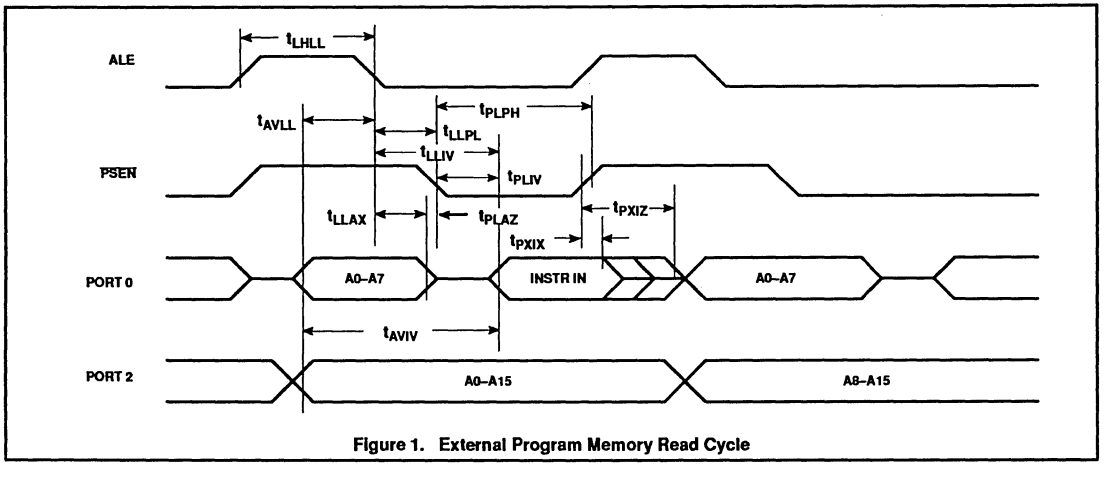


Figure 1. External Program Memory Read Cycle

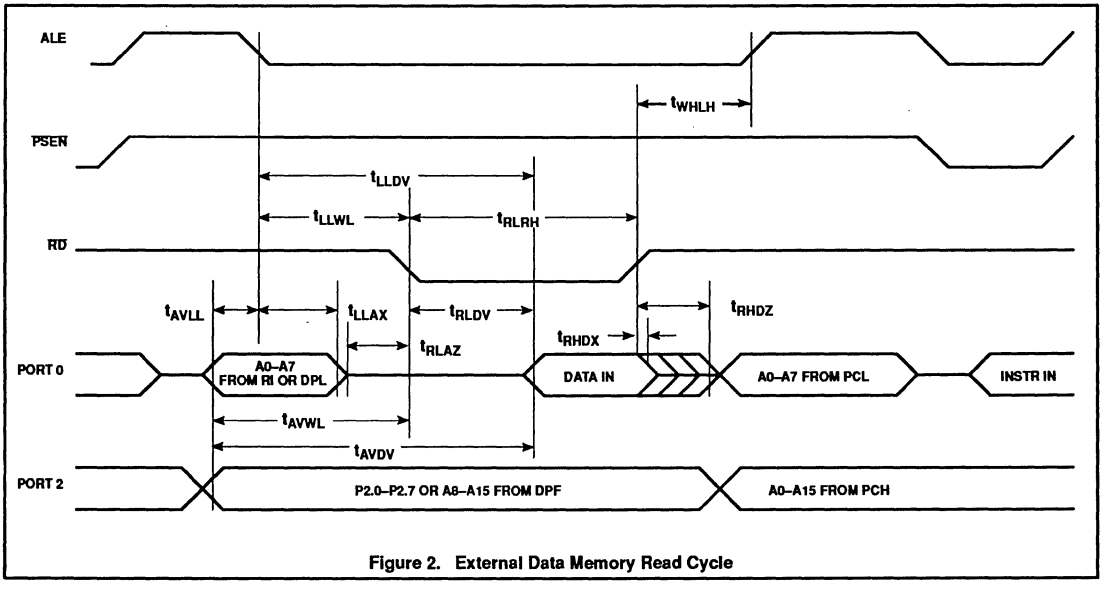


Figure 2. External Data Memory Read Cycle

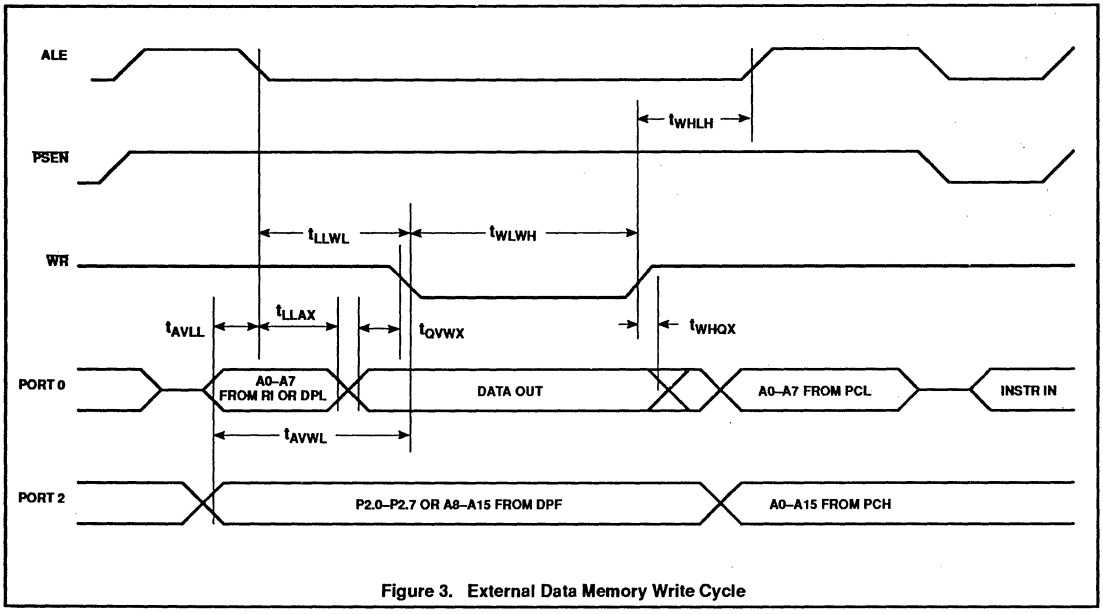


Figure 3. External Data Memory Write Cycle

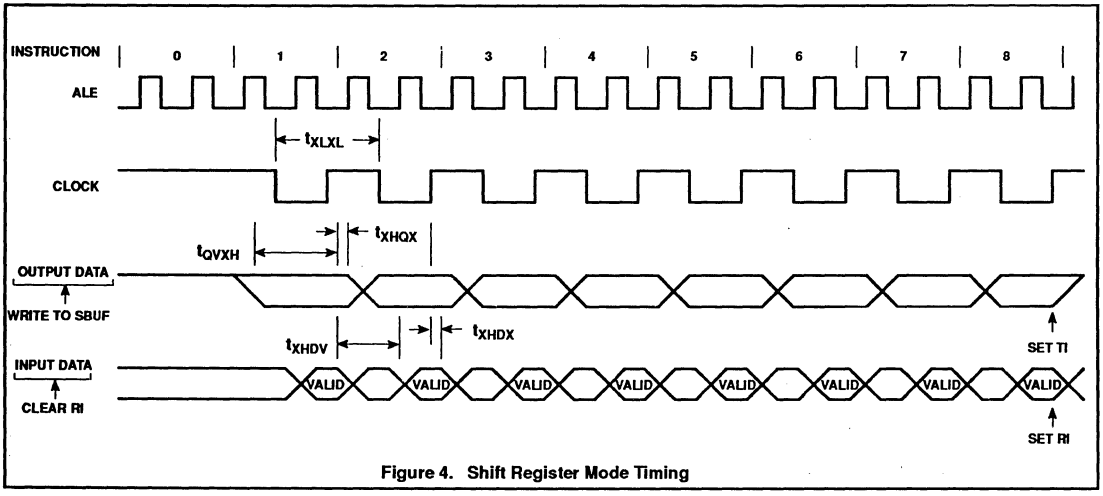


Figure 4. Shift Register Mode Timing

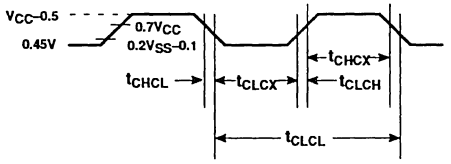
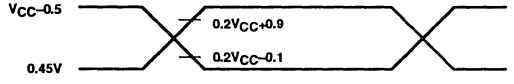
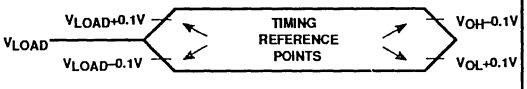


Figure 5. External Clock Drive



NOTE:
AC inputs during testing are driven at V_{CC} –0.5 for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at V_{IH} min for a logic '1' and V_{IL} for a logic '0'.

Figure 6. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 7. Float Waveform

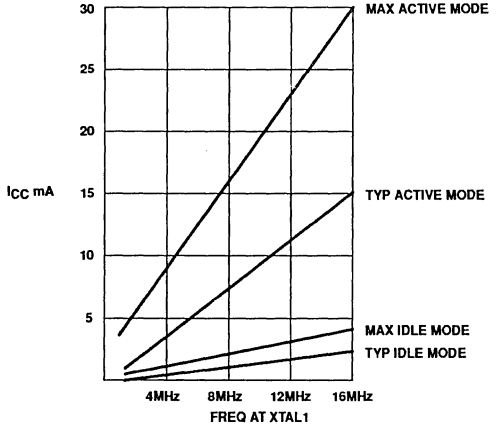


Figure 8. I_{CC} vs. FREQ
Valid only within frequency specifications of the device under test

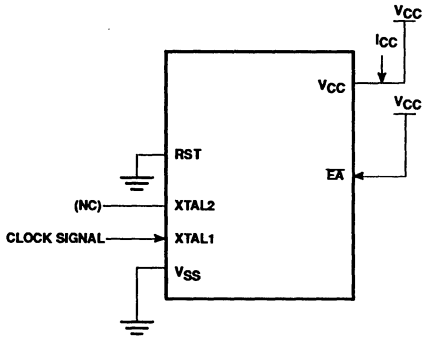


Figure 9. I_{CC} Test Condition, Active Mode
All other pins are disconnected

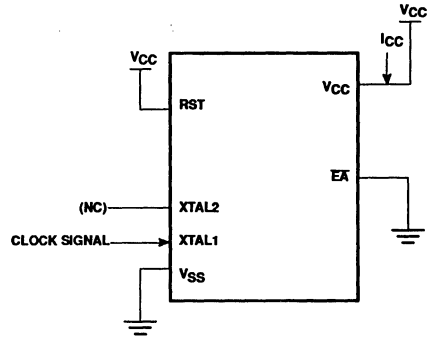


Figure 10. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

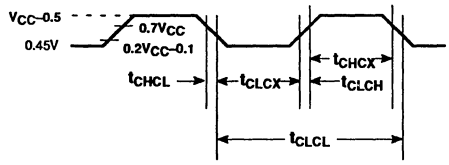


Figure 11. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

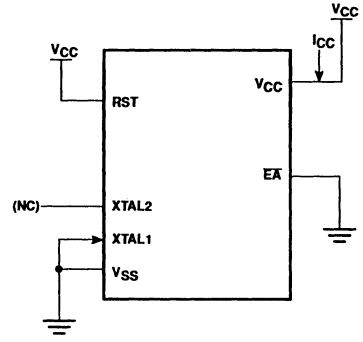


Figure 12. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

EPROM CHARACTERISTICS

To put the 87C575 in the EPROM programming mode, PSEN must be held high during power up, then driven low with reset active. The 87C575 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C575 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C575 manufactured by Philips.

Table 2 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the secuity bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 13 and 14. Figure 15 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 13. Note that the 87C575 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 13. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 2 and 3 specified in Table 2 are held at the 'Program Code Data' levels indicated in Table 2. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 14.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 1FH, using the 'Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the security bits, repeat the 25 pulse programming sequence using the 'Pgm Security Bit' levels. After one security bit is programmed, further programming of the code memory and encryption table is disabled. However, the other security bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If security bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 1 and 2 as shown in Figure 15. The other pins are held at the 'Verify Code Data' levels indicated in Table 2. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(B0H) = 97H indicates 87C575

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 2, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-s/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000µW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 2. EPROM Progamming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	0	0	1	1	0	0	0	0
Program code data	0	0	0*	V_{PP}	1	0	1	1
Verify code data	0	0	1	1	0	0	1	1
Pgm encryption table	0	0	0*	V_{PP}	1	0	1	0
Pgm security bit 1	0	0	0*	V_{PP}	1	1	1	1
Pgm security bit 2	0	0	0*	V_{PP}	1	1	0	0

NOTES:
1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.

* ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

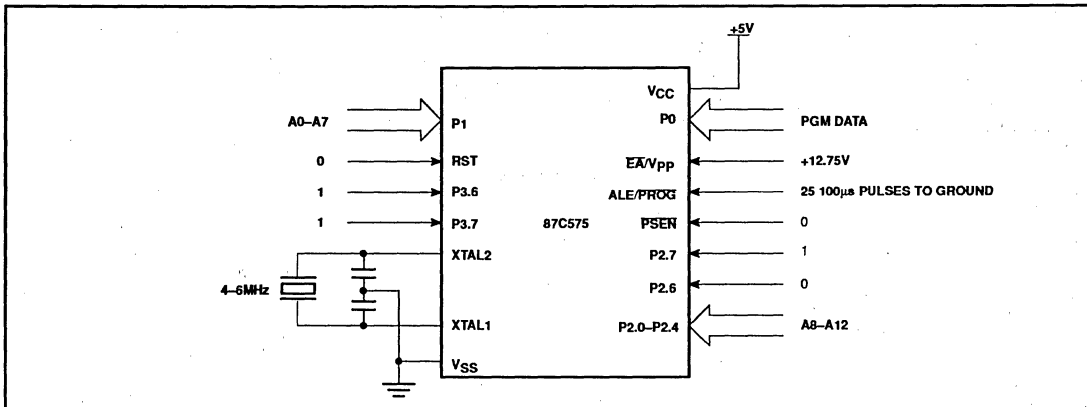


Figure 13. Programming Configuration

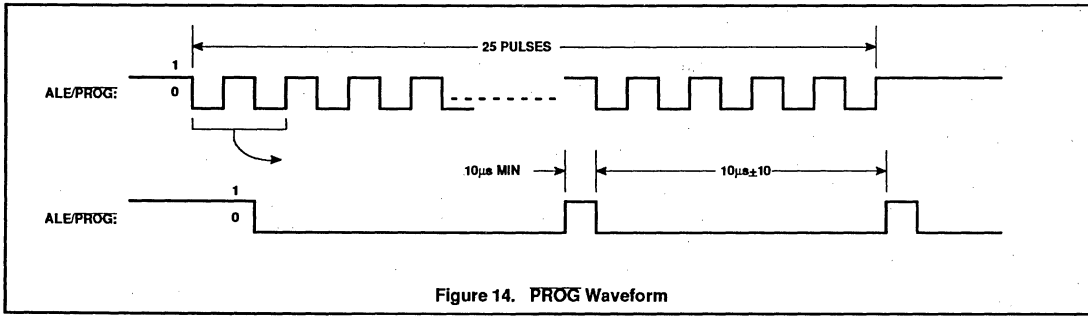


Figure 14. PROG Waveform

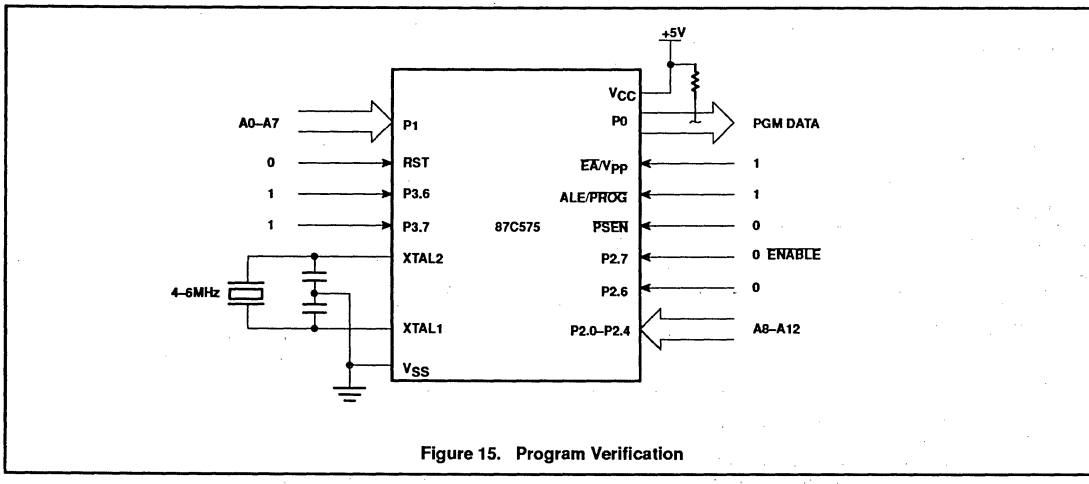


Figure 15. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_{amb} = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 16)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

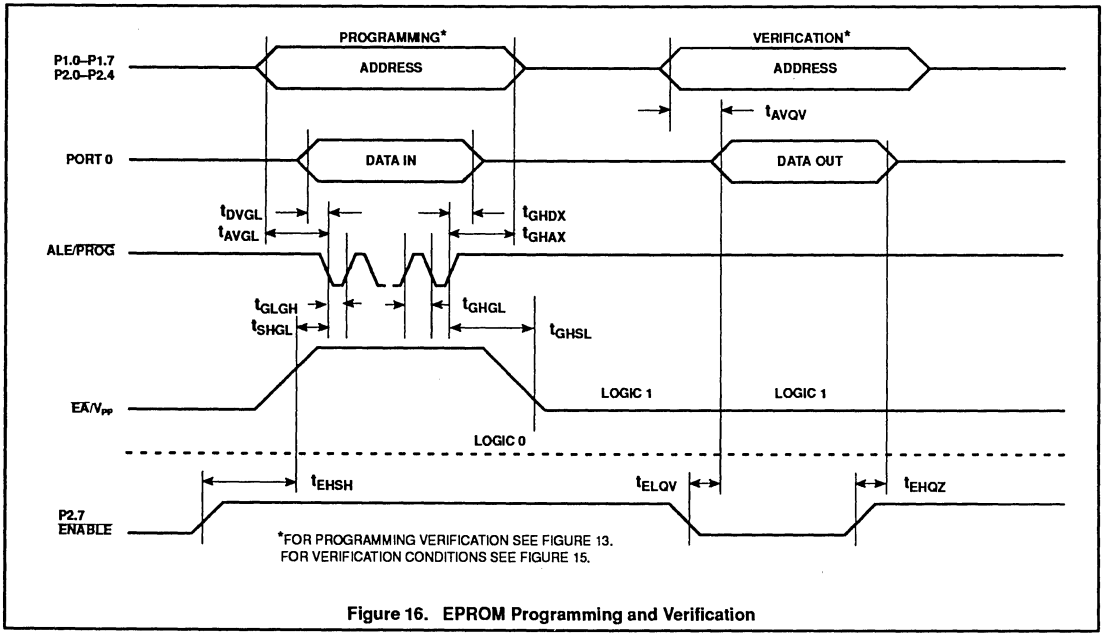


Figure 16. EPROM Programming and Verification

Single-chip 8-bit microcontroller with CAN controller

80C592/83C592/87C592

DESCRIPTION

The 80C592/83C592/87C592 (hereafter referred to generically as the 8XC592) is a stand-alone high-performance microcontroller designed for use in automotive and general industrial applications. In addition to the 80C51 standard features, this device provides a number of dedicated hardware functions for these applications. Three versions of this derivative will be offered:

- 83C592 (ROM version)
- 80C592 (ROMless version)
- 87C592 (EPROM/OTP version)

It combines the functions of the existing 8XC552 and the Philips CAN-Controller PCA82C200 (CAN: Controller Area Network) with the following enhanced features:

- 16K byte Program Memory
- 2 × 256 byte Data Memory
- DMA between CAN Transmit/Receive buffer and internal RAM

The temperature range includes –40°C to +85°C as well as automotive temperature range –40°C to +125°C for the ROM and ROMless version with a maximum clock frequency of 16MHz. The 87C592 has a temperature range of –40°C to +85°C.

The main differences to the 8XC552 microcontroller are:

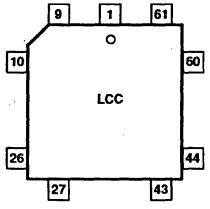
- a Can-controller substitutes the I^2C-serial interface
- 16K byte programmable ROM resp. EPROM instead of 8K byte
- additional 256 byte RAM.

The 8XC592 contains a 16k × 8 EPROM (87C592), ROM (83C592) program memory, a volatile 512 × 8 read/write data memory, a Controller Area Network (CAN) controller, six 8-bit I/O ports, one 8-bit input port, two 16-bit timer/event counters (identical to the timers of the 80C51), an additional 16-bit timer coupled to capture and compare latches, a 15-source, two-priority-level, nested interrupt structure, a 10-input ADC, a dual DAC pulse width modulated interface, two serial interfaces (UART and CAN), a "watchdog" timer and on-chip oscillator and timing circuits. For systems that require extra capability, the 8XC592 memory can be expanded externally using standard TTL compatible memories and logic.

FEATURES

- 80C51 core architecture
- 16k × 8 EPROM (87C592)
- 16k × 8 ROM (83C592)
- ROMless (80C592)
- 512 × 8 RAM, expandable externally to 64k bytes
- Two standard 16-bit timer/counters
- An additional 16-bit timer/counter coupled to four capture registers and three compare registers
- A 10-bit ADC with eight multiplexed analog inputs
- Two 8-bit resolution, pulse width modulation outputs
- 15 interrupt sources with 2 priority levels
- Five 8-bit I/O ports plus one 8-bit input port shared with analog inputs
- CAN controller with DMA transfer between internal data RAM and CAN registers
- 1 Mbit/s CAN-Controller with bus failure management facility
- $V_{DD}/2$ reference voltage
- Full-duplex UART compatible with the standard 80C51
- On-chip watchdog timer
- Extended temperature ranges (–40 to +125°C)
- OTP package available

PIN CONFIGURATION



Pin	Function	Pin	Function
1	P5.0/ADC0	35	V_{SS}
2	V_{DD}	36	P2.0/A08
3	STADC	37	P2.1/A09
4	$\overline{PWM0}$	38	P2.2/A10
5	$\overline{PWM1}$	39	P2.3/A11
6	$\overline{EW}$	40	P2.4/A12
7	P4.0/CMSR0	41	P2.5/A13
8	P4.1/CMRS1	42	P2.6/A14
9	P4.2CMSR2	43	P2.7/A15
10	P4.3/CMSR3	44	$\overline{PSEN}$
11	P4.4/CMSR4	45	ALE/$\overline{PROG}$
12	P4.5/CMSR5	46	$\overline{EA}/V_{PP}$
13	P4.6/CMT0	47	P0.7/AD7
14	P4.7/CMT1	48	P0.6/AD6
15	RST	49	P0.5/AD5
16	P1.0/CT0I/INT2	50	P0.4/AD4
17	P1.1/CT1I/INT3	51	P0.3/AD3
18	P1.2/CT2I/INT4	52	P0.2/AD2
19	P1.3/CT3I/INT5	53	P0.1/AD1
20	P1.4/T2	54	P0.0/AD0
21	P1.5/RT2	55	REF
22	CV_{SS}	56	CRX1
23	P1.6/CTX0	57	CRX0
24	P1.7/CTX1	58	AVref–
25	P3.0/RxD	59	AVref+
26	P3.1TxD	60	AV_{ss}
27	P3.2/$\overline{INT0}$	61	AV_{DD}
28	P3.3/$\overline{INT1}$	62	P5.7/ADC7
29	P3.4/T0	63	P5.6/ADC6
30	P3.5/T1	64	P5.5/ADC5
31	P3.6/$\overline{WR}$	65	P5.4/ADC4
32	P3.7/$\overline{RD}$	66	P5.3/ADC3
33	XTAL2	67	P5.2/ADC2
34	XTAL1	68	P5.1/ADC1

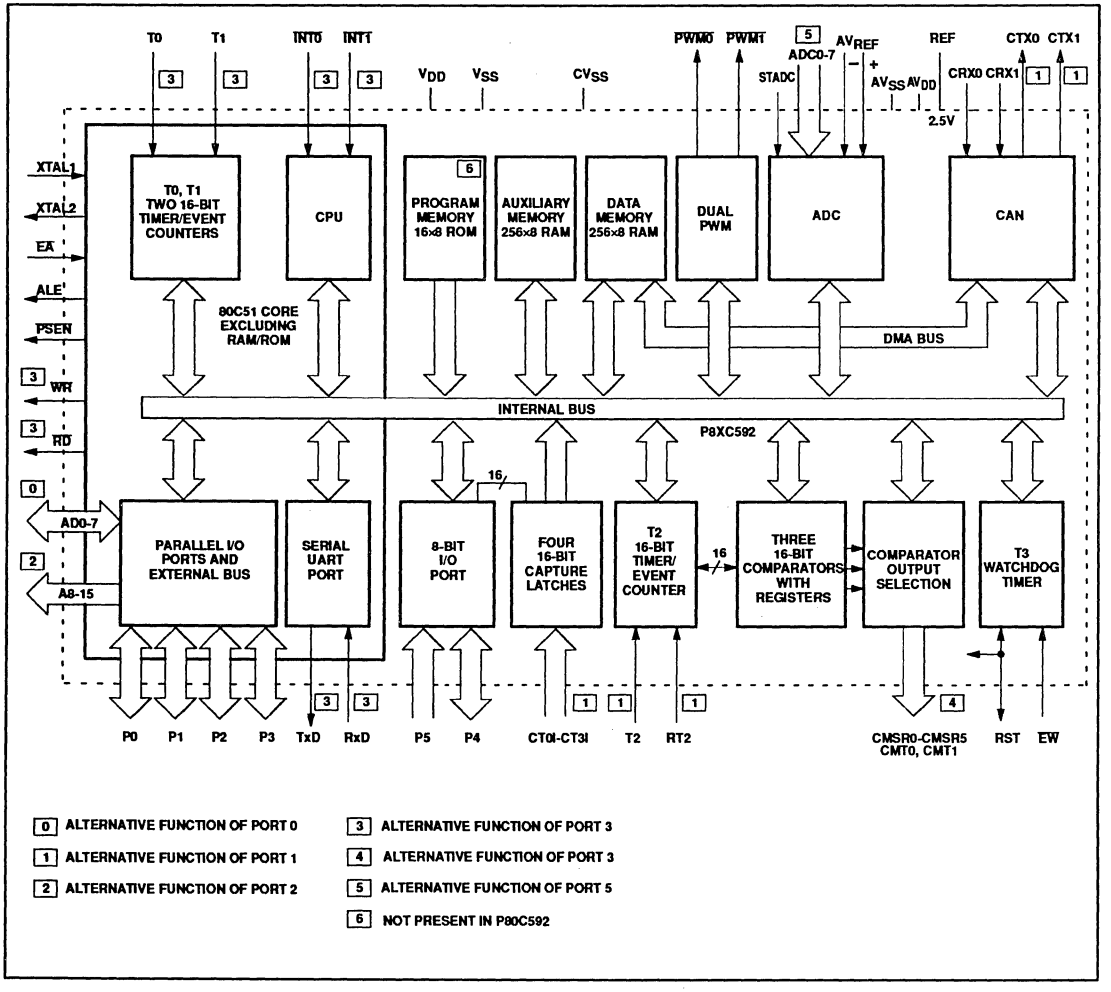
Single-chip 8-bit microcontroller with CAN controller

80C592/83C592/87C592

PART NUMBER SELECTION

ROMless	ROM	EPROM	TEMPERATURE °C AND PACKAGE	FREQUENCY
P80C592FFA	P83C592FFA	–	–40° to +85°C, PLCC 68	1.2 to 16MHz
P80C592FHA	P83C592FHA	–	–40° to +125°C, PLCC 68	1.2 to 16MHz
–	–	P87C592EFL	–40° to +85°C, ceramic window LCC	3.5 to 16MHz
–	–	P87C592EFA	–40° to +85°C, plastic LCC	3.5 to 16MHz

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{DD}	2	I	**Digital Power Supply:** +5V power supply pin during normal operation, idle and power-down mode.
STADC	3	I	**Start ADC Operation:** Input starting analog to digital conversion (ADC operation can also be started by software). This pin must not float.
$\overline{PWM0}$	4	O	**Pulse Width Modulation:** Output 0.
$\overline{PWM1}$	5	O	**Pulse Width Modulation:** Output 1.
$\overline{EW}$	6	I	**Enable Watchdog Timer:** Enable for T3 watchdog timer and disable power-down mode. This pin must not float.
P0.0-P0.7	54-47	I/O	**Port 0:** Port 0 is an 8-bit open-drain bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application it uses strong internal pull-ups when emitting 1s. Port 0 is also used to input the code byte during programming and to output the code byte during verification.
P1.0-P1.7	16-21, 23–24	I/O	**Port 1:** 8-bit I/O port. Alternate functions include:
	16-19	I	**CT0I-CT3I (P1.0-P1.3):** Capture timer input signals for timer T2.
	16-19	I	**INT2-INT5 (P1.0-P1.3):** External interrupts 2-5.
	20	I	**T2 (P1.4):** T2 event input. Rising edge triggered.
	21	I	**RT2 (P1.5):** T2 timer reset signal. Rising edge triggered.
	23	O	**CTX0 (P1.6):** CAN transmitter output 0.
	24	O	**CTX1 (P1.7):** CAN transmitter output 1. Port 1 is also used to input the lower order address byte during EPROM programming and verification. A0 is on P1.0, etc.
CV_{SS}	22	I	**CV_{SS}:** CAN transmitter driver ground.
P2.0-P2.7	36-43	I/O	**Port 2:** 8-bit quasi-bidirectional I/O port. Alternate function: High-order address byte for external memory (A08-A15). Port 2 is also used to input the upper order address during EPROM programming and verification. A8 is on P2.0, A9 on P2.1, through A12 on P2.4.
P3.0-P3.7	25-32	I/O	**Port 3:** 8-bit quasi-bidirectional I/O port. Alternate functions include:
	25		**RxD (P3.0):** Serial input port.
	26		**TxD (P3.1):** Serial output port.
	27		**$\overline{INT0}$ (P3.2):** External interrupt.
	28		**$\overline{INT1}$ (P3.3):** External interrupt.
	29		**T0 (P3.4):** Timer 0 external input.
	30		**T1 (P3.5):** Timer 1 external input.
	31		**$\overline{WR}$ (P3.6):** External data memory write strobe.
	32		**$\overline{RD}$ (P3.7):** External data memory read strobe.
P4.0-P4.7	7-14	I/O	**Port 4:** 8-bit quasi-bidirectional I/O port. Alternate functions include:
	7-12	O	**CMSR0-CMSR5 (P4.0-P4.5):** Timer T2 compare and set/reset outputs on a match with timer T2.
	13, 14	O	**CMT0, CMT1 (P4.6, P4.7):** Timer T2 compare and toggle outputs on a match with timer T2.
P5.0-P5.7	68-62, 1	I	**Port 5:** 8-bit input port. **ADC0-ADC7 (P5.0-P5.7):** Alternate function: Eight input channels to ADC.
RST	15	I/O	**Reset:** Input to reset the 8XC592. It also provides a reset pulse as output when the watchdog timer overflows or after a CAN wakeup from power-down.
XTAL1	34	I	**Crystal Pin 1:** Input to the inverting amplifier that forms the oscillator, and input to the internal clock generator. Receives the external clock signal when an external oscillator is used.
XTAL2	33	O	**Crystal Pin 2:** Output of the inverting amplifier that forms the oscillator. Left open-circuit when an external clock is used.
V_{SS}	35	I	**Digital ground.**
$\overline{PSEN}$	44	O	**Program Store Enable:** Active-low read strobe to external program memory.
ALE/$\overline{PROG}$	45	O	**Address Latch Enable:** Latches the low byte of the address during accesses to external memory. It is activated every six oscillator periods. During an external data memory access, one ALE pulse is skipped. ALE can drive up to eight LS TTL inputs and handles CMOS inputs without an external pull-up. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
$\overline{EA}/V_{PP}$	46	I	**External Access:** When $\overline{EA}$ is held at TTL level high, the CPU executes out of the internal program ROM provided the program counter is less than 16384. When $\overline{EA}$ is held at TTL low level, the CPU executes out of external program memory. $\overline{EA}$ is not allowed to float. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
REF	55	O	**REF:** $AV_{DD}/2$ reference voltage output or input, depending on CAN control register bits. If the internal reference is used, then REF should be connected to AV_{SS} through a 10nf (or greater) capacitor.
CRX1	56	I	**CRX1:** CAN receiver input line 1.
CRX0	57	I	**CRX0:** CAN receiver input line 0.
AV_{REF-}	58	I	**Analog to Digital Conversion Reference Resistor:** Low-end.
AV_{REF+}	59	I	**Analog to Digital Conversion Reference Resistor:** High-end.
AV_{SS}	60	I	**Analog Ground** (for ADC and CAN receiver)
AV_{DD}	61	I	**Analog Power Supply** (for ADC and CAN receiver)

NOTE:
1. To avoid "latch-up" effect at power-on, the voltage on any pin at any time must not be higher or lower than V_{DD} + 0.5V or V_{SS} – 0.5V, respectively.

The 8XC592 has the same operation as the 8XC552 for all features except the CAN interface and the AuxRAM. Please refer to the 8XC552 section in this data handbook for information on the PWM outputs, A/D converter, Timers 0, 1, or 2, the Watchdog Timer and the UART (SIO0).

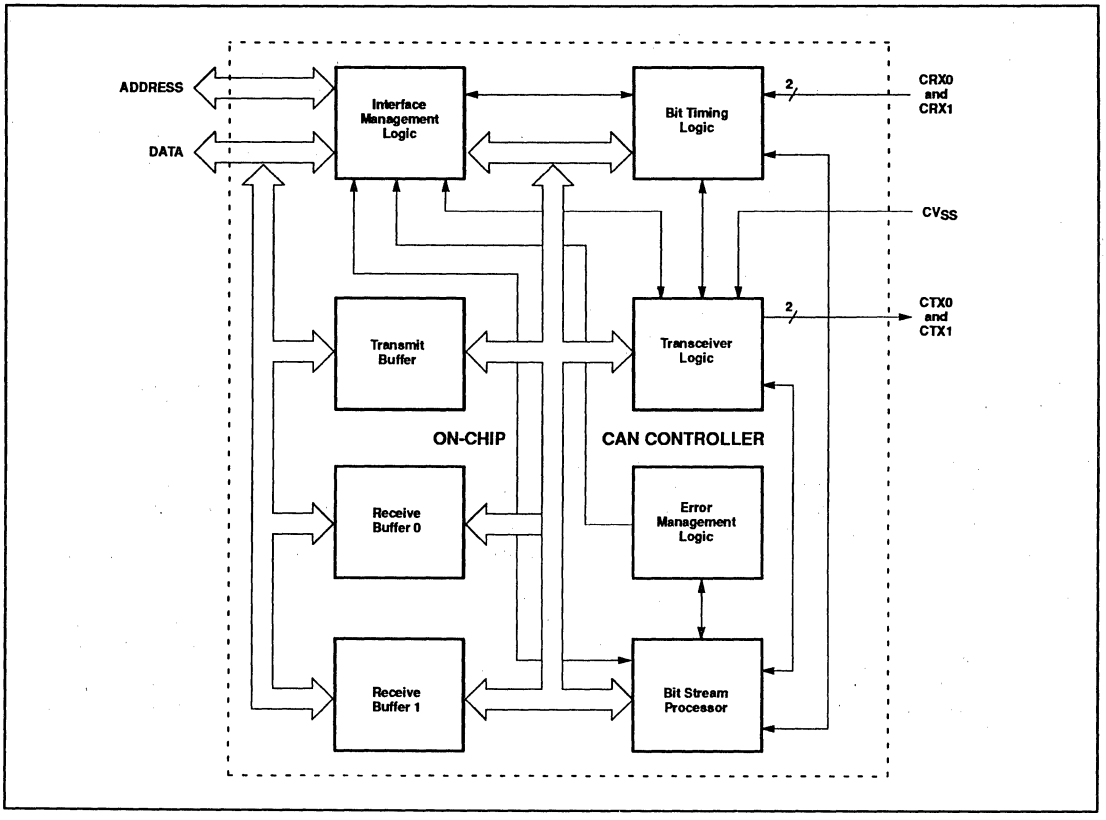
INTERNAL DATA MEMORY

The internal Data Memory is divided into three physically separated parts: the 256 byte of Main RAM, the 256 byte of AuxRAM, and the 128 byte special function area. these can be addressed each in a different way.

- Main RAM 0 to 127 can be addressed dirctly and indirectly as in the 80C51. Address pointers are R0 and R1 of the selected register bank.
- Main RAM 128 to 255 can only be addressed indirectly. Address pointers are R0 and R1 of the selected register bank.
- AuxRAM 0 to 255 is indirectly addressable in the same way as the external Data Memory with MOVX instructions. Address pointers are R0, R1 of the selected register bank and the DPTR. An access to AuxRAM 0 to 255 will not affect the ports P0, P2, P3.6 and P3.7 during internal program execution.

An access to external Data Memory locations higher than 255 will be performed with the MOVX @DPTR instructions in the same way as in the 80C51 structure, so with P0 and P2 as data/address bus and P3.6 and P.37 as write and read strobe signals. Note that these external Data Memory locations cannot be accessed with R0 or R1 as address pointer.

BLOCK DIAGRAM OF 8XC592 CAN CONTROLLER CIRCUITRY



CAN FUNCTIONAL DESCRIPTION

SIO1, CAN (Controller Area Network)

CAN is the definition of a high performance communication protocol for serial data communication. The 8XC592 on-chip CAN Controller is a full implementation of the CAN-protocol. With the 8XC592, powerful local networks can be built, both for automotive and general industrial environments. This results in a strongly reduced wiring harness and enhanced diagnostic and supervisory capabilities.

Features

- Multi-master architecture
- Bus access priority determined by the message identifier
- 2032 message identifier
- Guaranteed latency time for high priority messages
- Powerful error handling capability
- Data length from 0 up to 8 bytes
- Multicast and broadcast message facility
- Non-destructive bit-wise arbitration
- Non-return-to-zero (NRZ) coding/decoding with bit stuffing
- Programmable transfer rate (up to 1 Mbit/s)
- Programmable output driver configuration
- Suitable for use in a wide range of networks, including the SAE's network classes A, B and C
- DMA providing high-speed on-chip data exchange
- Bus failure management facility
- $AV_{DD/2}$ reference voltage

The CAN Controller meets the following automotive requirements:

- Short message length
- Guaranteed latency time* for urgent messages
- Bus access priority, determined by the message identifier
- Powerful error handling capability
- Configuration flexibility to allow area network expansion.

NOTE:

* The latency time defines the period between the initiation (Transmission Request) and the start of the transmission on the bus. The latency time strongly depends on a large variety of bus-related conditions. In the case of a message being transmitted on the bus and one distortion, the latency time can be up to 149 bit times (worst case). For more information, see the application note on bit timing.

CAN Functional Overview

The 8XC592 includes all hardware modules necessary to implement the transfer layer which represents the kernel of the CAN protocol. Refer to the block diagram (previous page) of the CAN controller portion of the 8XC592.

Interface Special Function Registers

The data transfer between the CPU and the CAN part of the 8XC592 is done via four SFRs:

CANADR	DBH
CANDAT	DAH
CANCON	D9H
CANSTA	D8H

To access a register of the CAN, you first have to store a 5-bit CAN address in CANADR and second read/write CANDAT. In the mode of auto-address-increment it is not necessary to write to CANADR (see section "Auto Address Increment"). The function of CANCON depends on its access direction. By writing CANCON you can access the CAN command register directly. A read access of CANCON shows which interrupts are enabled and the source of a CAN interrupt. After reading CANCON, all interrupt flags are reset. CANSTA (read access) is equivalent to the CAN Status register. It allows you to observe the status of the CAN. Writing CANSTA sets the RAM address for a DMA access. CANSTA is implemented as a bit addressable register for observing single status bits.

Accelerated Data Transfer

There are two features implemented in the 8XC592 to accelerate the exchange of data between CAN and CPU. The auto increment mode provides a fast stack-like reading and writing of the CAN registers, while the DMA mode allows you to transfer a whole message from or to the CPU internal main RAM.

Auto Address Increment: If the Auto Increment bit (CANADR.5) is high, the contents of CANADR is incremented automatically after any read or write access to CANDAT. For example, if you want to start a message, you have to address the Transmit Buffer only one time by writing a 10 into CANADR and setting the Auto Increment bit. After that, you may calculate databyte by databyte and move it to CANDAT.

Incrementing CANADR over 31 resets the Auto Increment bit automatically.

DMA: To enable the direct transfer of a CAN message between the RAM and Tx– or Rx–Buffer, you first have to write the RAM address (location of the first message byte, 0..FFH is addressable) into CANSTA; then you can address the Tx– (10) resp. Rx Buffer (≥ 20) and set the DMA bit (CANADR.7). This causes an automatic evaluation of the Data Length Code and then a transfer of the whole message via the DMA bus, which connects the CPU internal RAM with the CAN part.

The time needed for the DMA block move is up to two instruction cycles after the DMA bit was set; the transfer itself is done in the background, so the CPU can process the next instruction. An immediate access to the main internal RAM or to the CAN is not possible.

To order a Tx-DMA, CAN register 10 has to be addressed and the DMA bit set (8AH ≥ CANADR). The DLC is expected at the RAM address pointer plus 1, which was written into CANSTA before.

The Rx-DMA is very versatile. After setting the RAM address, you can store an arbitrary Rx address in CANADR, which causes a transfer of the message starting at the specified address from the Rx Buffer to the CPU RAM. This allows you, e.g., to collect only the data bytes of several messages in the RAM and process them combined.

After a successful DMA transfer, the DMA bit is reset.

NOTE: After a reset of the 8XC592, the DMA mode is disabled, the Auto Address Increment mode is enabled, and the address value contained by CANADR is four, so to point to the first register to be programmed after a reset (i.e., the Acceptance Code register).

Interface Management Logic (IML)

The IML interprets the commands from the CPU, controls the allocation of the message buffers Transmit Buffer (TBF), Receive Buffer 0 (RBF0), and Receive Buffer 1 (RBF1), and provides interrupts and status information to the CPU.

Transmit Buffer (TBF)

The TBF is an interface between the CPU and the Bit Stream Processor (BSP) and is able to store a complete message. The buffer is written by the CPU and read by the BSP. The TBF is 10 bytes long to hold the Descriptor (2 bytes) and the Data-Field (up to 8 bytes) of the message.

Receive Buffer (RBF0, RBF1)

The RBF is an interface between the BSP and the CPU and stores a message received from the busline. Once filled by BSP and allocated to the CPU by IML, the buffer cannot be used to store subsequently received messages until the CPU has (read and) released the buffer.

To reduce the requirements on the CPU, two receive buffers (RBF0, RBF1) are implemented. While one RBF is allocated to the CPU, the BSP may write to the other one. Both RBF0 and RBF1 are 10 bytes long to hold the Descriptor (2 bytes) and the Data-Field (up to 8 bytes) of the message.

Bit Stream Processor (BSP)

This is a sequencer controlling the data stream between transmit and receive buffers (parallel data) and the busline (serial data). The BSP contains the Acceptance Filter and also controls the TCL and the EML such that the processes of reception, arbitration, transmission, and error signaling are performed according to the protocol. The BSP provides signals to the IML indicating when a message has got acceptance, when a receive buffer contains a valid message, and also when the transmit butter is no longer required after a successful transmission.

Bit Timing Logic (BTL)

This block monitors the busline using the (built-in) Input Comparator and handles the busline-related bit timing.

The BTL synchronizes on a "recessive" to "dominant" busline transition at the beginning of a message (hard synchronization) and resynchronizes on further transitions during the reception of a message (soft synchronization).

The BTL also provides programmable time segments to compensate for the propagation delay times and phase shifts (e.g., due to oscillator drifts) and to define the sampling time and the number of samples (one or three) to be taken within a bit time.

Transceiver Logic (TCL)

The TCL controls the transmit output driver.

Error Management Logic (EML)

The EML is responsible for the error confinement of the transfer-layer modules. The EML gets error announcements from BSP and then informs the BSP, TCL, and IML about error statistics.

CONTROL SEGMENT AND MESSAGE BUFFER DESCRIPTION

The CAN Controller appears to the CPU as an on chip memory mapped peripheral, guaranteeing the independent operation of both parts.

Address Allocation

The address area of the CAN Controller consists of the Control Segment and the message buffers. The Control Segment is programmed during an initialization down-load in order to configure communication parameters (e.g., bit timing). The communication over the CAN-bus is also controlled via this segment by the CPU. A message which is to be transmitted, must be written to the Transmit Buffer. After a successful reception the CPU may read the message from the Receive Buffer and then release it for further use.

Control Segment Layout

The exchange of status, control and command signals between the CPU and the CAN Controller is performed in the control segment. The layout of this segment is shown in Figure 1. After an initial down-load, the contents of the registers Acceptance Code, Acceptance Mask, bus Timing 0 and 1 and Output Control should not be changed. These registers may only be accessed when the Reset Request bit in the Control Register is set HIGH (see "Control Register").

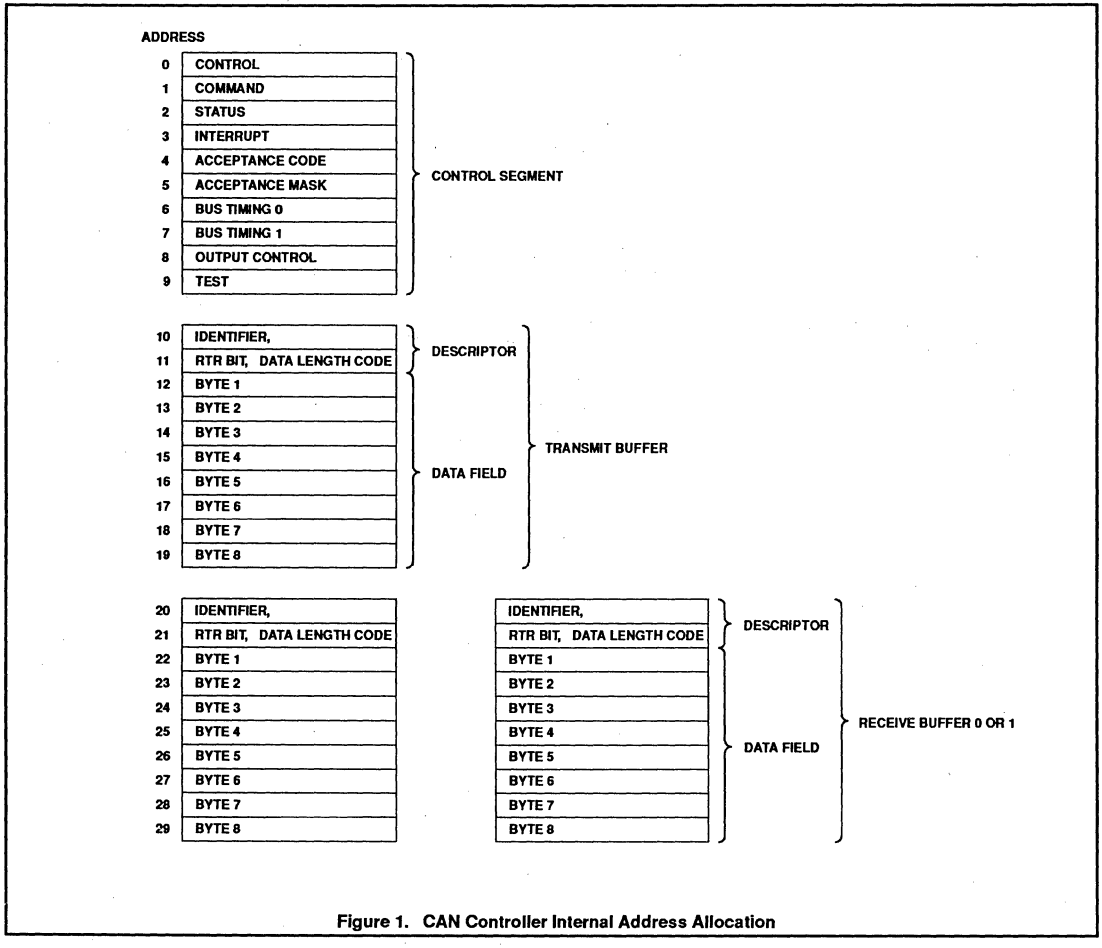


Figure 1. CAN Controller Internal Address Allocation

Table 1. CAN Registers

DESCRIPTION	ADDRESS	7 (MSB)	6	5	4	3	2	1	0 (LSB)
Control Segment									
Control Register	0	Test Mode	Synch	Reference Active	Overrun Interrupt Enable	Error Interrupt Enable	Transmit Interrupt Enable	Receive Interrupt Enable	Reset Request
Command Register	1	RX0 Active	RX1 Active	Wake-Up Mode	Sleep	Clear Overrun Status	Release Receive Buffer	Abort Transmission	Transmission Request
Status Register	2	Bus Status	Error Status	Transmit Status	Receive Status	Transmission Complete Status	Transmit Buffer Access	Data Overrun	Receive Buffer Status
Interrupt Register	3	reserved	reserved	reserved	Wake-Up Interrupt	Overrun Interrupt	Error Interrupt	Transmit Interrupt	Receive Interrupt
Acceptance Code Register	4	AC.7	AC.6	AC.5	AC.4	AC.3	AC.2	AC.1	AC.0
Acceptance Mask Register	5	AM.7	AM.6	AM.5	AM.4	AM.3	AM.2	AM.1	AM.0
Bus Timing Register 0	6	SJW.1	SJW.0	BRP.5	BRP.4	BRP.3	BRP.2	BRP.1	BRP.0
Bus Timing Register 1	7	SAM	TSEG2.2	TSEG2.1	TSEG2.0	TSEG1.3	TSEG1.2	TSEG1.1	TSEG1.0
Output Control Register	8	OCTP1	OCTN1	OCPOL1	OCTP0	COTN0	OCPOL0	OCMODE1	OCMODE0
Transmit Buffer									
Identifier	10	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3
RTR, Data Length Code	11	ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0
Bytes 1–8	12–19	Data	Data	Data	Data	Data	Data	Data	Data
Receive Buffer 0/1									
Identifier	20	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3
RTR, Data Length Code	21	ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0
Bytes 1–8	22–29	Data	Data	Data	Data	Data	Data	Data	Data

Control Register (CR) (Address 0)

The contents of the Control Register are used to change the behavior of the CAN Controller. Control bits may be set or reset by the CPU which uses the Control Register as a read/write memory.

Table 2. Description of the Control Register Bits

BIT	SYMBOL	CONTROL BIT	VALUE	COMMENTS
CR.0	RR	Reset Request[1]	HIGH (present)	Detection of a Reset Request results in the CAN Controller aborting the current transmission/reception of a message entering the reset state.
			LOW (absent)	On the HIGH-to-LOW transition of the Reset Request bit, the CAN Controller returns to its normal operating state.
CR.1	RIE	Receive Interrupt Enable	HIGH (enabled)	When a message has been received without errors, the CAN Controller transmits a Receive Interrupt signal to the CPU.
			LOW (disabled)	No transmission of the Receive Interrupt signal by the CAN Controller to the CPU.
CR.2	TIE	Transmit Interrupt Enable	HIGH (enabled)	When a message has been successfully transmitted or the transmit buffer is accessible again, (e.g., after an Abort Transmission command) the CAN Controller transmits a Transmit Interrupt signal to the CPU.
			LOW (disabled)	No transmission of the Transmit Interrupt signal by the CAN Controller to the CPU.
CR.3	EIE	Error Interrupt Enable	HIGH (enabled)	If the Error or Bus Status change (see status Register), the CPU receives an Error Interrupt signal.
			LOW (disabled)	The CPU receives no Error Interrupt signal.
CR.4	OIE	Overrun Interrupt Enable	HIGH (enabled)	If the Data Overrun bit is set (see Status Register), the CPU receives an Overrun Interrupt signal.
			LOW (disabled)	The CPU receives no Overrun Interrupt signal from the CAN Controller.
CR.5	RA	Reference Active[2]	HIGH (output)	The pin REF is an $AV_{DD/2}$ reference output.
			LOW (input)	A reference voltage may be input.
CR.6	S	Synch[2]	HIGH (2 edges)	Bus-line transitions from recessive-to-dominant and vice versa are used for resynchronization.
			LOW (1 edge)	Only transitions from recessive-to-dominant are used for resynchronization.
CR.7	–	RESERVED		

NOTES:

1. During an external reset (RST = HIGH) or when the Bus Status bit is set HIGH (Bus-Off), the IML forces the Reset Request HIGH (present). During an external reset the CPU cannot set the Reset Request bit LOW (absent). Therefore, after having set the Reset Request bit LOW (absent), the CPU must check this bit to ensure that the external reset pin is not being held HIGH (present). After the Reset Request bit is set LOW (absent) the CAN controller will wait for:
   - one occurance of the Bus-Free signal (11 recessive bits), if the preceding reset (Reset Request = HIGH) has been caused by an external reset or a CPU initiated reset.
   - 128 occurances of Bus-Free, if the preceding reset (Reset Request = HIGH) has been caused by a CAN Controller initiated Bus-Off, before re-entering the Bus-On mode.
   - When Reset Request is set HIGH (present), for whatever reason, the control, command, status and interrupt bits are affected, see Table 3. Only, when Reset Request is set HIGH (present) the registers at addresses 4 to 8 are accessible.
2. A modification of the bits Reference Active and Synch is only possible with Reset Request = HIGH (present). It is allowed to set these bits while Reset Request is changed from HIGH to LOW. After an external reset (pin RST = HIGH) the Reference Active bit is set HIGH (output), the Synch bit is undefined.

Table 3. Effects of Setting the Reset Request Bit HIGH (present)

TYPE	BIT	EFFECT
Control	Reference Active	HIGH (output), only after an external reset
Command	RX0 Active / RX1 Active	HIGH (RX0=CRX0, RX1=CRX1), only after an external reset
	Sleep	LOW (wake-up)
	Clear Overrun Status	HIGH (clear)
	Release Receive Buffer	HIGH (released)
	Abort Transmission	LOW (absent)
	Transmission Request	LOW (absent)
Status	Bus Status	LOW (Bus-On), only after an external reset
	Error Status	LOW (no error), only after an external reset
	Transmit Status	LOW (idle)
	Receive Status	LOW (idle)
	Transmission Complete Status	HIGH (complete)
	Transmit Buffer Access	HIGH (released)
	Data Overrun	LOW (absent)
	Receive Buffer Status	LOW (empty)
Interrupt	Overrun Interrupt	LOW (reset)
	Transmit Interrupt	LOW (reset)
	Receive Interrupt	LOW (reset)

Command Register (CMR)

A command bit initiates an action within the transfer layer of the CAN controller. the Command Register appears to the CPU as a read/write memory, except of the bits CMR.0 to CMR.3, which return HIGH if being read.

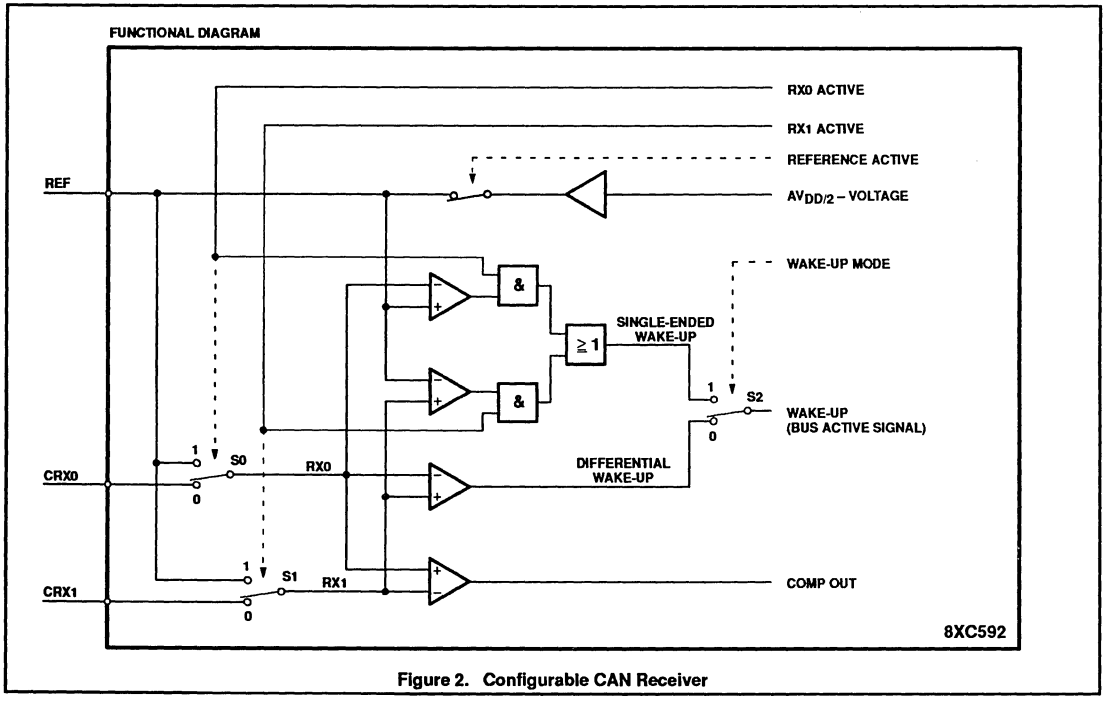


Figure 2. Configurable CAN Receiver

Table 4. Description of the Command Register Bits

BIT	SYMBOL	CONTROL BIT	VALUE		COMMENTS
CMR.0	TR	Transmission Request[1]	HIGH (present)		A message shall be transmitted.
			LOW (absent)		No action.
CMR.1	AT	Abort Transmission[2]	HIGH (present)		If not already in progress, a pending Transmission Request is cancelled.
			LOW (absent)		No action.
CMR.2	RRB	Release Receive Buffer[3]	HIGH (released)		The Receive Buffer attached to the CPU is released.
			LOW (no action)		No action.
CMR.3	COS	Clear Overrun[4]	HIGH (clear)		The Data Overrun status bit is set to LOW (see Status Register).
			LOW (no action)		No action.
CMR.4	SLP	Sleep[5]	HIGH (sleep)		The CAN Controller enters sleep mode if no CAN interrupt is pending and there is no bus activity.
			LOW (wake up)		The CAN Controller functions normally.
CMR.5	WUM	Wake-Up Mode[6]	HIGH (single ended)		The difference of the RX signals to the internal reference voltage $AV_{DD/2}$ is used for wake up.
			LOW (differential)		The differential signal between RX0 and RX1 is used for wake up.
CMR.6	RX1A	RX1 Active[7]	RX0 Active	RX1 Active	See Figure 2.
CMR.7	RX0A	RX0 Active[7]	1	1	RX0 = CRX0, RX1 = CRX1
			1	0	RX0 = CRX0, RX1 = $AV_{DD/2}$
			0	1	RX0 = $AV_{DD/2}$, RX1 = CRX1
			0	0	No action.

NOTES:

1. If the Transmission Request bit was set HIGH in a previous command, it cannot be cancelled by setting the Transmission Request bit LOW (absent). Cancellation of the requested transmission may be performed by setting the Abort Transmission bit HIGH (present).
2. The Abort Transmission bit is used when the CPU requires the suspension of the previously requested transmission, e.g., to transmit an urgent message. A transmission already in progress is not stopped. In order to see if the original message had been either transmitted successfully or aborted, the Transmission Complete Status bit should be checked. this should be done after the Transmit Buffer Access bit has been set HIGH (released) or a Transmit Interrupt has been generated (see Interrupt Register).
3. After reading the contents of the Receive Buffer (RBF0 or RBF1) the CPU must release this buffer by setting Release Receive Buffer bit HIGH (released). This may result in another message becoming immediately available.
4. This command bit is used to acknowledge the Data Overrun condition signaled by the Data Overrun status bit. It may be given or set at the same time as a Release Receive Buffer command bit.
5. The CAN Controller will enter sleep mode, if the Sleep bit is set HIGH (sleep), there is no bus activity and no interrupt is pending. A CAN Controller will wake up after the Sleep bit is set LOW (wake up) or when there is bus activity. On wake up, a Wake-Up Interrupt (see Interrupt Register) is generated (see "Power Reduction Modes"). A CAN Controller which is sleeping and then awoken by bus activity will not be able to receive this message until it detects a Bus-Free signal. The Sleep bit, if being read, reflects the status of the CAN Controller.
6. The Wake-Up Mode bit should be set at the same time as the Sleep bit. The differential wake up mode is useful if both bus wires are fully functioning; it minimizes the probability of wake ups due to noise. The single ended wake up mode is recommended if a wake up must be possible even if one bus wire is already or may become disturbed (See Figure 2).
7. The RX0/RX1 Active bits, if being read, reflect the status of the respective switches (See Figure 2). It is recommended to change the switches only during the reset state (Reset Request is HIGH).

Status Register (SR)

The contents of the Status Register reflects the status of the CAN Controller. The Status Register appears to the CPU as a read only memory.

Table 5. Description of the Status Register Bits

BIT	SYMBOL	STATUS BIT	VALUE	COMMENTS
SR.0	RBS	Receive Buffer Status[1]	HIGH (full)	This bit is set when a new message is available.
			LOW (empty)	No message has become available since the last Release Receive Buffer command bit was set.
SR.1	DO	Data Overrun[2]	HIGH (overrun)	This bit is set HIGH (overrun) when both Receive Buffers are full and the first byte of another message should be stored.
			LOW (absent)	No data overrun has occurred since the Clear Overrun command was given.
SR.2	TBS	Transmit Buffer Access[3]	HIGH (released)	The CPU may write a message into the TBF.
			LOW (locked)	The CPU cannot access the Transmit Buffer. A message is either waiting for transmission or is in the process of being transmitted.
SR.3	TCS	Transmit Complete Status[3]	HIGH (complete)	Last requested transmission has been successfully completed.
			LOW (incomplete)	Previously requested transmission is not yet completed.
SR.4	RS	Receive Status[4]	HIGH (receive)	The CAN Controller is receiving a message.
			LOW (idle)	No message is received.
SR.5	TS	Transmit Status[4]	HIGH (transmit)	The CAN Controller is transmitting a message.
			LOW (idle)	No message is transmitted.
SR.6	ES	Error Status	HIGH (error)	At least one of the Error Counters has reached the CPU Warning limit.
			LOW (ok)	Both Error Counters have not reached the warning limit.
SR.7	BS	Bus Status[5]	HIGH (Bus-Off)	The CAN Controller is not involved in bus activities.
			LOW (Bus-On)	The CAN Controller is involved in bus activities.

NOTES:

1. If the command bit Release Receive Buffer is set HIGH (released) by the CPU, the Receive Buffer Status bit is set LOW (empty) by IML. When a new message is stored in any of the receive buffers, the Receive Buffer Status bit is set HIGH (full) again.
2. If Data Overrun = HIGH (overrun) is detected, the currently received message is dropped. A transmitted message, granted acceptance, is also stored in a Receive Buffer. This occurs because it is not known if the CAN Controller will lose arbitration and so become a receiver of the message. If no Receive Buffer is available, Data Overrun is signaled. However, this transmitted and accepted message does neither cause a Receive Interrupt nor set the Receive Buffer Status bit to HIGH (full). Also, a Data Overrun does not cause the transmission of an Overload Frame.
3. If the CPU tries to write to the Transmit Buffer when the Transmit Buffer Access bit is LOW (locked), the written bytes will not be accepted and will be lost without this being signaled. The Transmission Complete Status bit is set LOW (incomplete) whenever the Transmission Request bit is set HIGH (present). If an Abort Transmission command is issued, the Transmit Buffer will be released. If the message, which was requested and then aborted, was not transmitted, the Transmission Complete Status bit will remain LOW.
4. If both the Receive Status and Transmit Status bits are LOW (idle) the CAN-bus is idle.
5. When the Bus Status bit is set HIGH (Bus-Off), the CAN Controller will set the Reset Request bit HIGH (present). It will stay in this state until the CPU sets the Reset Request bit LOW (absent). Once this is completed, the CAN Controller will wait the minimum protocol-defined time (128 occurrences of the Bus-Free signal) before setting the Bus Status bit LOW (Bus-On), the Error Status bit LOW (ok), and resetting the Error Counters.

Interrupt Register (IR)

The Interrupt Register allows the identification of an interrupt source. When one or more bits of this register are set, a CAN interrupt (SIO1) will be indicated to the CPU. All bits are reset by the CAN Controller after this register is read by the CPU. This register appears to the CPU as a read only memory.

Table 6. Description of the Interrupt Register Bits

BIT	SYMBOL	INTERRUPT BIT	VALUE	COMMENTS
IR.0	RI	Receive Interrupt[1]	HIGH (set)	This bit is set when a new message is available in the Receive Buffer and the Receive Interrupt Enable bit is HIGH (enabled).
			LOW (reset)	Receive Interrupt bit is automatically reset by a read access of Interrupt Register by the CPU.
IR.1	TI	Transmit Interrupt	HIGH (set)	This bit is set on a change of the Transmit Buffer Access from LOW to HIGH (released) and Transmit Interrupt Enable is HIGH (enabled).
			LOW (reset)	Transmit Interrupt bit will be reset after a read access of the Interrupt Register by the CPU.
IR.2	EI	Error Interrupt	HIGH (set)	This bit is set on a change of either the Error Status or Bus Status bits (see Status Register) if the Error Interrupt Enable is HIGH (enabled).
			LOW (reset)	The Error Interrupt bit is reset by a read access of the Interrupt Register by the CPU.
IR.3	OI	Overrun Interrupt[2]	HIGH (set)	This bit is set HIGH, if both Receive Buffers contain a message and the first byte of another message should be stored (passed acceptance), and the Overrun Interrupt Enable is HIGH (enabled).
			LOW (reset)	Overrun Interrupt bit is reset by a read access of the Interrupt register by the CPU.
IR.4	WUI	Wake-Up Interrupt	HIGH (set)	The Wake-Up Interrupt bit is set HIGH when the sleep mode is left (see Command Register).
			LOW (reset)	Wake-Up Interrupt bit is reset by a read access of the Interrupt Register by the CPU.
IR.5	–	RESERVED		
IR.6	–	RESERVED		
IR.7	–	RESERVED		

NOTES:
1. Receive Interrupt bit (if enabled) and Receive Buffer Status bit (see Status Register) are set at the same time.
2. Overrun Interrupt bit (if enabled) and Data Overrun bit (see Command Register) are set at the same time.

Acceptance Code Register (ACR)

The Acceptance Code Register is part of the acceptance filter of the CAN Controller. This register can be accessed (read/write), if the Reset Request bit is set HIGH (present). When a message is received which passes the acceptance test and if there is an empty Receive Buffer, then the respective Descriptor and Data Field (see Figure 1) are sequentially stored in this empty buffer. In the case that there is no empty Receive buffer, the Data Overrun bit is set HIGH (overrun), see Status and Interrupt Register. When the complete message has been correctly received, the following occurs:

- The Receive Buffer Status bit is set HIGH (full)
- If the Receive Interrupt Enable bit is set HIGH (enabled), the receive Interrupt is set HIGH (set).

The Acceptance Code bits (AC.7–AC.0) and the eight most significant bits of the message's Identifier (ID.10–ID.3) must be equal to those bit positions which are marked relevant by the Acceptance Mask bits (AM.7–AM.0). If the following equation is satisfied, acceptance is given:

[(ID.10 . . ID.3) EQUAL (AC.7 . . AC.0)] or (AM.7 . . AM.0) = 1111 1111_b

Acceptance Code Register Bits

ACR	ADDRESS 4						
7	6	5	4	3	2	1	0
AC.7	AC.6	AC.5	AC.4	AC.3	AC.2	AC.1	AC.0

During transmission of a message which passes the acceptance test, the message is also written to its own Receive Buffer. If no Receive Buffer is available, Data Overrun is signaled because it is not known at the start of a message whether the CAN Controller will lose arbitration and so become a receiver of the message.

Acceptance Mask Register (AMR)

The Acceptance Mask Register is part of the acceptance filter of the CAN Controller. This register can be accessed (read/write) if the Reset Request bit is set HIGH (present). The Acceptance Mask Register qualifies which of the corresponding bits of the acceptance code are "relevant" or "don't care" for acceptance filtering.

Acceptance Mask Register Bits

AMR	ADDRESS 5						
7	6	5	4	3	2	1	0
AM.7	AM.6	AM.5	AM.4	AM.3	AM.2	AM.1	AM.0

Description of the Acceptance Mask Register Bits

ACCEPTANCE MASK BIT	VALUE	COMMENTS
AM.7 to AM.0	HIGH (don't care)	This bit position is "don't care" for the acceptance of a message.
	LOW (relevant)	This bit position is "relevant" for acceptance filtering.

Bus Timing Register 0 (BTR 0)

The contents of Bus Timing Register 0 defines the values of the Baud Rate Prescaler (BRP) and the Synchronization Jump Width (SJW). This register can be accessed (read/write) if the Reset Request bit is set HIGH (present).

Bus Timing Register 0 Bits

BTR0	ADDRESS 6						
7	6	5	4	3	2	1	0
SJW.1	SJW.0	BRP.5	BRP.4	BRP.3	BRP.2	BRP.1	BRP.0

Baud Rate Prescaler (BRP)

The period of the system clock t_{SCL} is programmable and determines the individual bit timing. The system clock is calculated using the following equation:

$t_{SCL} = 2t_{CLK}$ (32BRP.5 + 16BRP.4 + 8BRP.3 + 4BRP.2 + 2BRP.1 + BRP.0 +1)

t_{CLK} = time period of the 8XC592 oscillator.

Synchronization Jump Width (SJW)

To compensate for phase shifts between clock oscillators of different bus controllers, any bus controller must resynchronize on any relevant signal edge of the current transmission. The synchronization jump width defines the maximum number of clock cycles a bit period may be shortened or lengthened by one resynchronization:

$t_{SJW} = t_{SCL}$ (2SJW.1 + SJW.0 + 1)

Bus Timing Register 1 (BTR1)

The contents of Bus Timing Register 1 defines the length of the bit period, the location of the sample point and the number of samples to be taken at each sample point. This register can be accessed (read/write) if the Reset Request bit is set HIGH (present).

Bus Timing Register 1 Bits

BTR1	ADDRESS 7			
7	6	5	4	
SAM	TSEG2.2	TSEG2.1	TSEG2.0	
	3	2	1	0
	TSEG1.3	TSEG1.2	TSEG1.1	TSEG1.0

Sampling (SAM)

BIT	VALUE	COMMENTS
SAM	HIGH (3 samples)	3 samples are taken.
	LOW (1 sample)	The bus is sampled once.

SAM = LOW is recommended for high speed buses (SAE class C), while SAM = HIGH is recommended for slow/medium speed buses (class A and B) where filtering of spikes on the bus-line is beneficial (see "Bit Timing Restrictions").

Time Segment 1 (TSEG1) and Time Segment 2 (TSEG2)

TSEG1 and TSEG2 determine the number of clock cycles per bit period and the location of the sample point:

$t_{TSEG1} = t_{SCL}$ (8TSEG1.3 + 4TSEG1.2 + 2TSEG.1 + TSEG1.0 + 1)

$t_{TSEG2} = t_{SCL}$ (4TSEG2.2 + 2TSEG2.1 + TSEG2.0 + 1)

For further information on bus timing see Bus Timing Register 0 and "Bus Timing/ Synchronization".

Output Control Register (OCR)

The Output Control Register allows, under software control, the set-up of different output driver configurations. This register can be accessed (read/write) if the Reset Request bit is set HIGH (present).

Output Control Register Bits

OCR	ADDRESS 8			
7	6	5	4	3
OCTP1	OCTN1	OCPOL1	OCTP0	OCTN0
	2	1		0
	OCPOL0	OCMODE1		OCMODE0

If the CAN Controller is in the sleep mode (Sleep = HIGH) a recessive level is output on the CTX0 and CTX1 pins. If the CAN Controller is in the reset state (Reset Request = HIGH) the output drivers are floating.

Normal Output Mode

In Normal Output Mode the bit sequence (TXD) is sent via CTX0 and CTX1. The voltage levels on the output driver pins CTX0 and CTX1 depend on both the driver characteristic programmed by OCTPx, OCTNx (float, pull-up, pull-down, push-pull) and the output polarity programmed by OCPOLx (see Figure 3).

Clock Output Mode

For the CTX0 pin this is the same as in Normal Output Mode. However, the data stream to CTX1 is replaced by the transmit clock (TXCLK). The rising edge of the transmit clock (non-inverted) marks the beginning of a bit period. The clock pulse width is t_{SCL}.

Bi-phase Output Mode

In contrast to Normal Output Mode the bit representation is time variant and toggled. If the bus controllers are galvanically decoupled from the bus-line by a transformer, the bit stream is not allowed to contain a DC component. This is achieved by the following scheme. During recessive bits all outputs are de-activated (floating). Dominant bits are sent alternatingly on CTX0 and CTX1, i.e., the first dominant bit is sent on CTX0, the second is sent on CTX1, and the third one is sent on CTX0 again, etc.

Test Output Mode

For the CTX0 pin this is the same as in Normal Output Mode. To measure the delay time of the transmitter and receiver this mode connects the output of the input comparator (COMP OUT) with the input of the output driver CTX1. This mode is used for production testing only.

The following two tables, Table 7 and Table 8, show the relationship between the bits of the Output Control Register and the two serial output pins CTX0 and CTX1 of the 8XC592 – CAN Controller, connected to the serial bus (see Block Diagram, page 484).

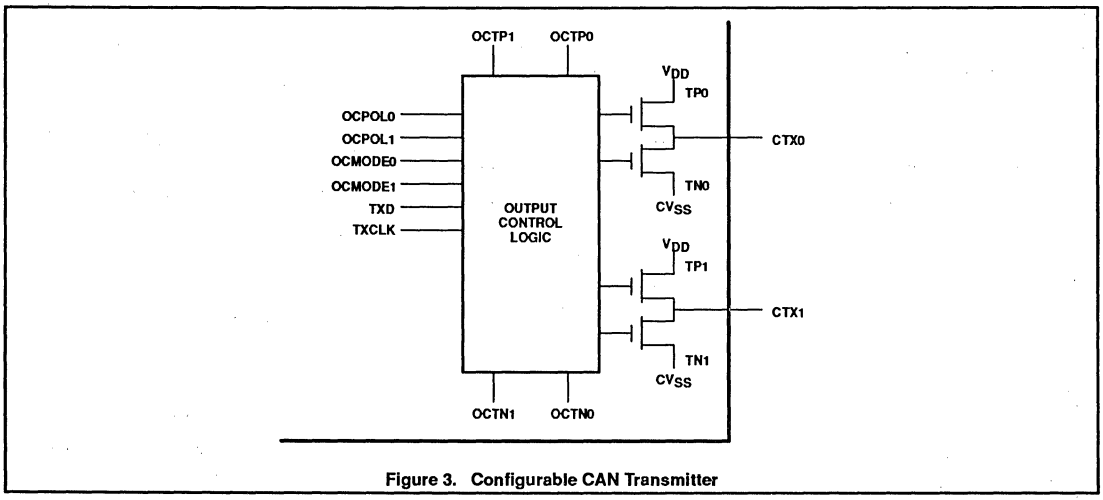


Figure 3. Configurable CAN Transmitter

Table 7. Description of the Output Mode Bits

OCMODE1	OCMODE0	DESCRIPTION
1	0	Normal output Mode; CTX0, CTX1: bit sequence (TXD; note 1)
1	1	Clock output Mode; CTX0: bit sequence, CTX1: bus clock (TXCLK)
0	0	Bi-phase output Mode
0	1	Test output Mode; CTX0: bit sequence, CTX1: COMP OUT

NOTE:
1. TXD is the data bit to be transmitted.

Table 8. Output Pin Set-Up

DRIVE	OCTPx	OCTNx	OCPOLx	TXD	TPx	TNx	CTXx
Float	0	0	0	0	Off	Off	float
	0	0	0	1	Off	Off	float
	0	0	1	0	Off	Off	float
	0	0	1	1	Off	Off	float
Pull-down	0	1	0	0	Off	On	LOW
	0	1	0	1	Off	Off	float
	0	1	1	0	Off	Off	float
	0	1	1	1	Off	On	LOW
Pull-up	1	0	0	0	Off	Off	float
	1	0	0	1	On	Off	HIGH
	1	0	1	0	On	Off	HIGH
	1	0	1	1	Off	Off	float
Push/Pull	1	1	0	0	Off	On	LOW
	1	1	0	1	On	Off	HIGH
	1	1	1	0	On	Off	HIGH
	1	1	1	1	Off	On	LOW

NOTES:
1. TPx is the on-chip output transistor x, connected to V_{DD}; x = 0 or 1.
2. TNx is the on-chip output transistor x, connected to CV_{SS}; x = 0 or 1.
3. CTXx is the serial output level on CTX0 or CTX1. It is required that the output level on the CAN bus is dominant with TXD = 0 and recessive with TXD = 1.

TRANSMIT BUFFER LAYOUT

The global layout of the Transmit Buffer is shown in Figure 1. This buffer serves to store a message from the CPU to be transmitted by the CAN Controller. It is subdivided into Descriptor and Data Field. the Transmit Buffer can be written to and read from by the CPU.

DESCRIPTOR

Descriptor Byte 1 (DSCR1)

DSCR1	ADDRESS 10						
7	6	5	4	3	2	1	0
ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3

Descriptor Byte 2 (DSCR2)

DSCR2	ADDRESS 11						
7	6	5	4	3	2	1	0
ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0

Identifier (ID)

The Identifier consists of 11 bits (ID.10 to ID.0) ID.10 is the most significant bit, which is transmitted first on the bus during the arbitration process. the Identifier acts as the message's name, used in a receiver for acceptance filtering, and also determines the bus access priority during the arbitration process. The lower the binary value of the Identifier, the higher the priority. this is due to the larger number of leading dominant bits during arbitration.

Remote Transmission Request Bit (RTR)

Description of the RTR Bit

BIT	VALUE	COMMENTS
RTR	HIGH (remote)	Remote Frame will be transmitted by the CAN Controller.
	LOW (data)	Data Frame will be transmitted by the CAN Controller.

Data Length Code (DLC)

The number of bytes (Data Byte Count) in the Data Field of a message is coded by the Data Length Code. at the start of a Remote Frame transmission, the Data Length Code is not considered due to the RTR bit being HIGH (remote). This forces the number of transmitted/received data bytes to be 0. Nevertheless, the Data Length code must be specified correctly to avoid bus errors, if two CAN-controllers start a Remote Frame transmission simultaneously.

The range of the Data Byte Count is 0 to 8 bytes and coded as follows:

Data Byte Count = 8DLC.3 + 4DLC.2 + 2DLC.1 + DLC.0

For reasons of compatibility, no Data Byte Counts other than 0,1,2,... and 8 should be used.

Data Field

The number of transferred data bytes is determined by the Data Length Code. the first bit transmitted is the most significant bit of data byte 1 at address 12.

RECEIVE BUFFER LAYOUT

The layout of the Receive Buffer and the individual bytes correspond to the definitions given for the Transmit Buffer layout, except that the addresses start at 20 instead of 10 (see Figure 1).

Table 9. The Special Function Registers Between CPU and CAN

SFR	ADR	ACS	MSB 7	6	5	4	3	2	1	LSB 0
CANADR	D8h	RW	DMA	reserved[3]	Auto Inc	CANA4	CANA3	CANA2	CANA1	CANA0
CANDAT	DAh	RW	CAND7	CAND6	CAND5	CAND4	CAND3	CAND2	CAND1	CAND0
CANCON[1]	D9h	R	reserved[3]	reserved[3]	reserved[3]	Wake Up Interrupt	Overrun Interrupt	Error Interrupt	Transmit Interrupt	Receive Interrupt
		W	RX0 Active	RX1 Active	Wake Up Mode	Sleep	Clear Overrun	Release RxBuf	Abort Transm	Transm Request
CANSTA[1,2]	D8h	R	Bus Status	Error Status	Transmit Status	Receive Status	TxComp1 Status	TxBuf Access	Data Overrun	RxBuf Status
		W	RAMA7	RAMA6	RAMA5	RAMA4	RAMA3	RAMA2	RAMA1	RAMA0

NOTES:
1. Do not use a RMW instruction.
2. The bit addresses of CANSTA (7:0) are DFH . . D8H.
3. Reserved bits are read as HIGH.

HANDLING OF THE CPU-CAN INTERFACE

Via the four special registers CANADR, CANDAT, CANCON and CANSTA, the CPU has access to the CAN Controller and also to the DAM-logic. Note that CANCON and CANSTA have different meanings for a read and write access.

CANADR

The five least significant bits CANADR.4 down to CANADR.0 (CANA4 . . . CANA0) define the address of one of the CAN Controller internal registers to be accessed via CANDAT. For instance, after an external hardware (e.g., power-on) reset CANADR contains the value 64h, and hence the CPU accesses (read/write) the Acceptance Code register of the CAN Controller, via the Special Function Register CANDAT. CANADR also controls the auto address increment mode via bit CANADR.5 (AutoInc) and the DMA-logic via bit CANADR.7 (DMA). CANADR is implemented as a read/write register.

CANDAT

The Special Function Register CANDAT appears as a port to the CAN Controller's internal register (memory location) being selected by CANADR. Reading or writing CANDAT is effectively an access to that CAN Controller internal register, which is selected by CANADR. CANDAT is implemented as a read/write register.

CANCON

When reading CANCON the Interrupt Register of the CAN Controller is accessed, while writing to CANCON means an access to the Command Register. CANCON is implemented as a read/write register.

CANSTA

Reading CANSTA is an access to the Status Register of the CAN Controller. Writing to CANSTA sets the address of the on-chip Main RAM (internal data memory) for a subsequent DMA transfer. CANSTA is implemented as a bit-addressable read/write register.

Auto Address Increment

With the auto address increment mode a fast stack-like reading and writing of CAN Controller internal registers is provided. If the bit CANADR.5 (AutoInc) is high, the content of CANADR is incremented automatically after any read or write access to CANDAT. For instance, loading a message into the Transmit Buffer can be done by writing 2AH into CANADR and then moving byte by byte of the message to CANDAT.

Incrementing CANADR beyond $xx111111_b$ resets the bit CANADR.5 (AutoInc) automatically (CANADR = $xx000000_b$).

High Speed DMA

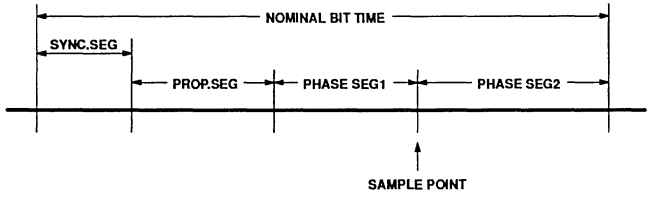
The DMA-logic allows to transfer a complete message (up to 10 bytes) between CAN Controller and Main RAM in 2 instruction cycles at maximum; up to 4 bytes are transferred in 1 instruction cycle. The performance of the CPU is strongly enhanced because this very fast transfer is don in the background. A DMA transfer is achieved by first writing the RAM address (0 . . FFH) into CANSTA and then setting the TX- or RX-Buffer address in CANADR and the bit CANADR.7 (DMA) simultaneously; the RAM address points to the location of the first byte to be transferred. Setting the DMA bit causes an automatic evaluation of the Data Length Code and then the transfer; for a TX-DMA transfer the Data Length Code is expected at the location "RAM address + 1".

In order to program a TX-DMA transfer, the value 8AH (address 10) has to be written into CANADR. Then a complete message, consisting of the 2-byte Descriptor and the Data Field (0 . . . 8 bytes), starting at location "RAM address" is transferred to the TX-Buffer.

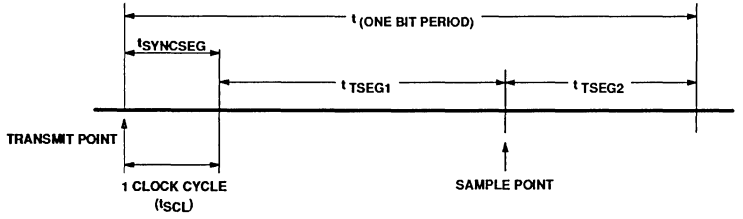
The RX-DMA transfer is very versatile. By writing a value in the range of 94H (address 20) up to 9DH (address 29) of CANADR the whole or a part of the received message, starting at the specified address, is transferred to the internal data memory. This allows e.g. to transfer the bytes of the Data Field only.

After a successful DMA transfer the DMA- (and Auto Inc-) bit is reset.

During a DMA transfer the CPU can process the next instruction. However, an access to the data memory, CANADR, CANDAT, CANCON or CANSTA is not allowed. After having set the DMA-bit, every interrupt is disabled until the end of the transfer. Note, that disadvantageous programming may lead to an interrupt response time of at most 10 instruction cycles. The shortest interrupt response time is achieved by using 2 consecutive 1-cycle instructions directly after setting the DMA-bit. During the reset state (bit Reset Request is HIGH) a DMA transfer is not possible.



a) As defined by the CAN protocol.



a) As implemented in the 8XC592's on-chip CAN Controller.

Figure 4. Bit Period

BUS TIMING / SYNCHRONIZATION

The Bus Timing Logic (BTL) monitors the serial bus-line via the on-chip input comparator and performs the following functions:

- Monitors the serial bus-line level
- Adjusts the sample point, within a bit period (programmable)
- Samples the bus-line level using majority logic (programmable, 1 or 3 samples)
- Synchronization to the bit stream:
  - Hard synchronization at the start of a message
  - Resynchronization during transfer of a message

The configuration of the BTL is performed during the initialization of the CAN Controller. The BTL uses the following three registers:

- Control register (Synch)
- Bus Timing Register 0
- Bus Timing Register 1

Bit Timing

A bit period is built up from a number of system clock cycles (t_{SCL}) (see "Bus Timing Register 0"). One bit period is the result of the addition of the programmable segments TSEG1 and TSEG2 and the general segment SYNCSEG.

Synchronization Segment (SYNCSEG)
The incoming edge of a bit is expected during this state; this state corresponds to one system clock cycle ($1 \times t_{SCL}$).

Time Segment 1 (TSEG1)
This segment determines the location of the sampling point within a bit period, which is at the end of TSEG1. TSEG1 is programmable from 1 to 16 system clock cycles.

The correct location of the sample point is essential for the correct functioning of a transmission. The following points must be taken into consideration:

- A Start-Of-Frame causes all CAN Controllers to perform a "hard synchronization" on the first recessive-to-dominant edge. During arbitration, however, several CAN Controllers may simultaneously transmit. Therefore it may require twice the sum of bus-line, input comparator and the output driver delay times until the bus is stable. This is the propagation delay time.
- To avoid sampling at an incorrect position, it is necessary to include an additional synchronization buffer on both sides of the sample point. The main reasons for incorrect sampling are:
  - Incorrect synchronization due to spikes on the bus-line;
  - Slight variations in the oscillator frequency of each CAN Controller in the network, which results in a phase error.
- Time Segment 1 consists of the segment for compensation of propagation delays and the synchronization buffer segment directly before the sample point (see Figure 4).

Time Segment 2 (TSEG2)
This time segment provides:

- Additional time at the sample point for calculation of the subsequent bit levels (e.g., arbitration);
- Synchronization buffer segment directly after the sample point.

TSEG2 is programmable from 1 to 8 system clock cycles.

Synchronization Jump Width (SJW)
SJW defines the maximum number of clock cycles (t_{SCL}) a period may be reduced or increased by one resynchronization. SJW is programmable from 1 to 4 system clock cycles.

Propagation Delay Time (t_{PROP})
The propagation delay time is calculated by summing the maximum propagation delay times of the physical bus, the input comparator and the output driver. The resulting sum is multiplied by 2 and then rounded up to the nearest multiple of t_{SCL}.

$t_{PROP} = 2 \times$ (physical bus delay + input comparator delay + output driver delay)

Bit Timing Restrictions
Restrictions on the configuration of the bit timing are based on internal processing. The restrictions are:

- $t_{TSEG2} \geq 2t_{SCL}$
- $t_{TSEG2} \geq t_{SJW}$
- $t_{TSEG1} \geq t_{TSEG2}$
- $t_{TSEG1} \geq t_{SJW} + t_{PROP}$

The three sample mode (SAM = 1) has the effect of introducing a delay of one system clock cycle on the bus-line. This must be taken into account for the correct calculation of TSEG1 and TSEG2:

- $t_{TSEG1} \geq t_{SJW} + t_{PROP} + 2t_{SCL}$
- $t_{TSEG2} \geq 3t_{SCL}$

Synchronization

Synchronization is performed by a state machine which compares the incoming edge with its actual bit timing and adapts the bit timing by hard synchronization or resynchronization.

Hard Synchronization
This type of synchronization occurs only at the beginning of a message. The CAN Controller synchronizes on the first incoming recessive-to-dominant edge of a message (being the leading edge of a message's Start-Of-Frame bit).

Resynchronization
Resynchronization occurs during the transmission of a message's bit stream to compensate for:

- Variations in individual CAN Controller oscillator frequencies;
- Changes introduced by switching from one transmitter to another (e.g., during arbitration).

As a result of resynchronization, either t_{TSEG1} may be increased by up to a maximum of t_{SJW}, or t_{TSEG2} may be decreased by up to a maximum of t_{SJW}:

- $t_{TSEG1} \leq t_{SCL}$ [(TSEG1 + 1) + (SJW + 1)]
- $t_{TSEG2} \geq t_{SCL}$ [(TSEG2 + 1) – (SJW + 1)]

NOTE: TSEG1, TSEG2 and SJW are the programmed numerical values.

The phase error (e) of an edge is given by the position of the edge relative to SYNCSEG, measured in system clock cycles (t_{SCL}). The value of the phase error is defined as:

- $e = 0$, if the edge occurs within SYNCSEG
- $e > 0$, if the edge occurs within TSEG1
- $e < 0$, if the edge occurs within TSEG2.

The effect of resynchronization is:

- The same as that of a hard synchronization, if the magnitude of the phase error (e) is less or equal to the programmed value of t_{SJW};
- To increase a bit period by the amount of t_{SJW}, if the phase error is positive and the magnitude of the phase error is larger than t_{SJW};
- To decrease a bit period by the amount of t_{SJW}, if the phase error is negative and the magnitude of the phase error is larger than t_{SJW}.

Synchronization Rules
The synchronization rules are as follows:

- Only one synchronization within one bit time used.
- An edge is used for synchronization only if the value detected at the previous sample point differs from the bus value immediately after the edge.
- Hard synchronization is performed whenever there is a recessive-to-dominant edge during Bus-Idle.
- All other edges (recessive-to-dominant and optionally dominant-to-recessive edges if the Synch bit is set HIGH) which are candidates for resynchronization will be used with the following exception:
  - A transmitting CAN Controller will not perform a resynchronization as a result of a recessive-to-dominant edge with positive phase error, if only these edges are used for resynchronization. this ensures that the delay times of the output driver and input comparator do not cause a permanent increase in the bit time.

CAN-PROTOCOL DESCRIPTION

Frame Types

The 8XC592's CAN Controller supports the four different CAN protocol frame types for communication:

- Data Frame, to transfer data
- Remote Frame, request for data
- Error Frame, globally signals a (locally) detected error condition
- Overload Frame, to extend delay time of subsequent frames (an Overload Frame is not initiated by the 8XC592 CAN Controller.

Bit Representation

There are two logical bit representations used in the CAN protocol:

- A recessive bit on the bus-line appears only if all connected CAN Controllers send a recessive bit at that moment
- Dominant bits always overwrite recessive bits, i.e., the resulting bit level on the bus-line is dominant.

Data Frame

A Data Frame carries data from a transmitting CAN Controller to one or more receiving ones. A Data Fram is composed of seven different bit-fields:

- Start-Of-Frame
- Arbitration Field
- Control Field
- Data Field (may have a length of zero)
- CRC Field
- Acknowledge Field
- End-Of-Frame

Start-of-Frame Bit

Signals the start of a Data Frame or Remote Frame. It consists of a single dominant bit used for hard synchronization of a CAN Controller in receive mode.

Arbitration Field

Consists of the message Identifier and the RTR bit. In the case of simultaneous message transmissions by two or more CAN Controllers the bus access conflict is solved by bit-wise arbitration, which is active during the transmission of the Arbitration Field.

Identifier

This 11–bit field is used to provide information about the message, as well as the bus access priority. It is transmitted in the order ID.10 to ID.0 (LSB). The situation that the seven most significant bits (ID.10 to ID.4) are all recessive must not occur.

An Identifier does not define which particular CAN Controller will receive the frame because a CAN-based communication network does not differentiate between a point-to-point, multicast or broadcast communication.

RTR Bit

A CAN Controller, acting as a receiver for certain information may initiate the transmission of the respective data by transmitting a Remote Frame to the network, addressing the data source via the Identifier and setting the RTR bit HIGH (remote; recessive bus level). If the data source simultaneously transmits a Data Frame containing the requested data, it uses the same Identifier. No bus access conflict occurs due to the RTR bit being set LOW (data; dominant bus level) in the Data Frame.

Control Field

This field consists of six bits. It includes two reserved bits (for future expansions of the CAN protocol), transmitted with a dominant bus level, and is followed by the Data Length Code (4 bits). The number of bytes (destuffed; number of data bytes to be transmitted/received) in the Data Field is indicated by the Data Length Code. Admissible values of the Data Length Code, and hence the number of bytes in the (destuffed) Data Field, are (0,1, ..., 8). A logic 0 (logic 1) in the Data Length Code is transmitted as dominant (recessive) bus level, respectively.

Data Field

The data, stored within the Data Field of the transmit buffer, are transmitted according to the Data Length Code. Conversely, data of a received Data Frame will be stored in the Data Field of a Receive Buffer. The Data Field can contain from 0 up to 8 bytes. The most significant bit of the first data byte (lowest address) is transmitted/received first.

Cyclic Redundancy Code Field (CRC)

The CRC Field consists of the CRC Sequence (15 bits) and the CRC Delimiter (1 recessive bit). The Cyclic Redundancy Code (CRC) encloses the destuffed bit stream of the Start-of-Frame, Arbitration Field, Control Field, Data Field and CRC Sequence. The most significant bit of the CRC Sequence is transmitted/received first. This frame check sequence, implemented in the CAN Controller is derived from a cyclic redundancy code best suited for frames with a total bit count of less than 127 bits, CRC Error detection. With Start-of-Frame (dominant bit) included in the code word, any rotation of the code word can be detected by the absence of the CRC Delimiter (recessive bit).

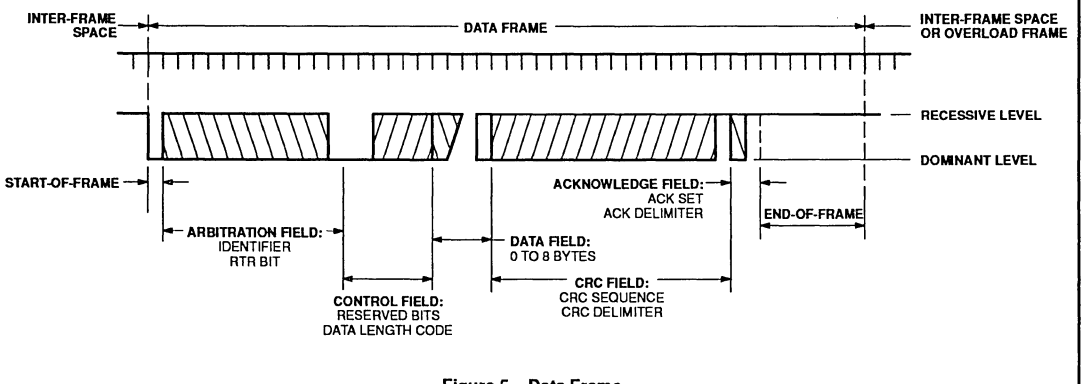


Figure 5. Data Frame

Acknowledge Field (ACK)
The Acknowledge Field consists of two bits, the Acknowledge Slot and the Acknowledge Delimiter, which are transmitted with a recessive level by the transmitter of the Data Frame. All CAN Controllers having received the matching CRC Sequence, report this by overwriting the transmitter's recessive bit in the Acknowledge Slot with a dominant bit. Thereby a transmitter, still monitoring the bus level recognizes that at least one receiver within the network has received a complete and correct message (i.e., no error was found). The Acknowledge Delimiter (recessive bit) is the second bit of the Acknowledge Field. As a result, the Acknowledge Slot is surrounded by two recessive bits: the CRC Delimiter and the Acknowledge Delimiter.

All nodes within a CAN network may use all the information coming to the network by all CAN Controllers (shared memory concept). Therefore, acknowledgement and error handling are defined to provide all information in a consistent way throughout this shared memory. Hence, there is no reason to discriminate different receivers of a message in the acknowledge field. If a node is disconnected from the network due to bus failure, this particular node is no longer part of the shared memory. To identify a "lost node" additional and application specific precautions are required.

End-Of-Frame
Each Data Frame or Remote Frame is delimited by the End-of-Frame bit sequence which consists of seven recessive bits (exceeds the bit stuff width by two bits). Using this method a receiver detects the end of a frame independent of a previous transmission error because the receiver expects all bits up to the end of the CRC Sequence to be coded by the method of bit-stuffing. The bit-stuffing logic is deactivated during the End-of-Frame sequence.

Remote Frame

A CAN Controller acting as a receiver for certain information may initiate the transmission of the respective data by transmitting a Remote Frame to the network, addressing the data source via the Identifier and setting the RTR bit HIGH (remote; recessive bus level). The Remote Frame is similar to the Data Frame with the following exceptions:

- RTR bit is set HIGH
- Data Length Code is ignored
- No Data Field contained

Note that the value of the Data Length Code should be the one of the corresponding Data Frame, although it is ignored for a Remote Frame.

A Remote Frame is composed of six different bit fields:

- Start-of-Frame
- Arbitration Field
- Control Field
- CRC Field
- Acknowledge Field
- End-of-Frame

Error Frame

The Error Frame consists of two different fields. The first field is accomplished by the superimposing of Error Flags contributed from different CAN Controllers.

The second field is the Error Delimiter.

Error Flag
There are two forms of an Error Flag:

- Active Error Flag, consists of six consecutive dominant bits
- Passive Error Flag, consists of six consecutive recessive bits unless it is overwritten by dominant bits from other CAN Controllers.

An error-active CAN Controller detecting an error condition signals this by transmission of an Active Error Flag. This Error Flag's form violates the bit-stuffing rule applied to all fields, from Start-of-Frame to CRC Delimiter, or destroys the fixed form of the fields Acknowledge Field or End-of-Frame. Consequently, all other CAN Controllers detect an error condition and start transmission of an Error Flag. Therefore the sequence of dominant bits, which can be monitored on the bus, results from a superposition of different Error Flags transmitted by individual CAN Controllers. The total length of this sequence varies between six (minimum) and twelve (maximum) bits.

An error-passive CAN Controller detecting an error condition tries to signal this by transmission of a Passive Error Flag. The error-passive CAN Controller waits for six consecutive bits with identical polarity, beginning at the start of the Passive Error Flag. The Passive Error Flag is complete when these six identical bits have been detected.

Error Delimiter
The Error Delimiter consists of eight recessive bits and has the same format as the Overload Delimiter. After transmission of an Error Flag, each CAN Controller monitors the bus-line until it detects a transition from a dominant-to-recessive bit level. At this point in time, every CAN Controller has finished sending its Error Flag and has additionally sent the first out of the 8 recessive bits of the Error Delimiter. Afterwards all CAN Controllers transmit the remaining recessive bits. After this event and an Intermission Field all error-active CAN Controllers within the network can start a transmission simultaneously.

If a detected error is signaled during transmission of a Data Frame or Remote Frame, the current message is spoiled and a retransmission of the message is initiated.

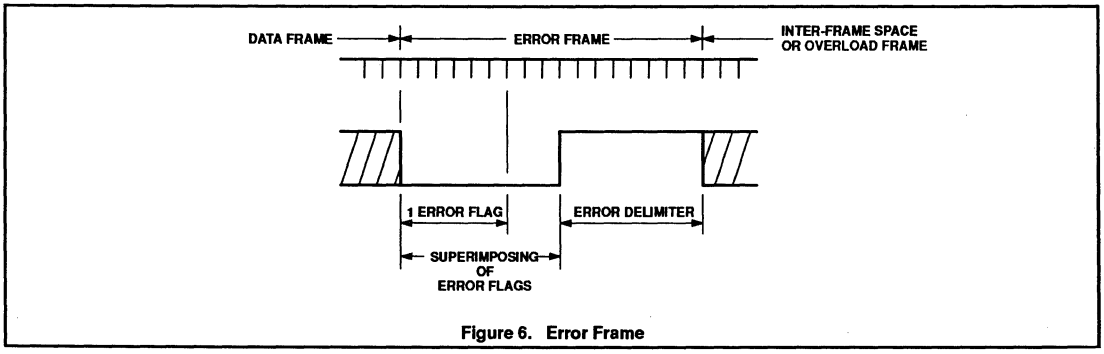


Figure 6. Error Frame

If a CAN Controller monitors any deviation of the Error Frame, a new Error Frame will be transmitted. Several consecutive Error Frames may result in the CAN Controller becoming error-passive and leaving the network unblocked.

In order to terminate an Error Flag correctly, an error-passive CAN Controller requires the bus to be Bus-Idle for at least three bit periods (if there is a local error at an error-passive-receiver). Therefore a CAN bus should not be permanently loaded.

Overload Frame

The Overload Frame consists of two fields, the Overload Flag and the Overload Delimiter. There are two conditions in the CAN protocol which lead to the transmission of an Overload Flag:

- Condition 1; receiver circuitry requires more time to process the current data before receiving the next frame (receiver not ready)
- Condition 2; detection of a dominant bit during Intermission Field.

The transmission of an Overload Frame may only start:

- Condition 1; during the first bit period of an expected Intermission Field
- Condition 2; one bit period after detecting the dominant bit during Intermission Field

The 8XC592's on chip CAN Controller will never initiate transmission of a condition 1-Overload Frame and will only react on a transmitted condition 2 Overload Frame, according to the CAN protocol. No more than two Overload Frames are generated to delay a Data Frame or a Remote Frame. Although the overall from of the Overload Frame corresponds to that of the Error Frame, an Overload Frame does not initiate or require the retransmission of the preceding frame.

Overload Flag

The Overload Flag consists of six dominant bits and has a similar format to the Error Flag.

The Overload Flag's form corrupts the fixed form of the Intermission Field. All other CAN Controllers detecting the overload condition also transmit an Overload Flag (condition 2).

Overload Delimiter

The Overload Delimiter consists of eight recessive bits and takes the same form as the Error Delimiter. After transmission of an Overload Flag, each CAN Controller monitors the bus-line until it detects a transition from a dominant-to-recessive bit level. At this point in time, every CAN Controller has finished sending its Overload Flag and all CAN Controllers start simultaneously transmitting seven more recessive bits.

Inter-Frame Space

Data Frames and Remote Frames are separated from preceding frames (all types) by an Inter-Frame Space, consisting of an Intermission Field and a Bus-Idle. Error-passive CAN Controllers also send a Suspend Transmission after transmission of a message. Overload Frames and Error Frames are not preceded by an Inter-Frame Space.

Intermission Field

The Intermission Field consists of three recessive bits. During an Intermission period, no frame transmissions will be started by the 8XC592's on chip CAN Controller. An Intermission is required to have a fixed time period to allow a CAN Controller to execute internal processes prior to the next receive or transmit task.

Bus-Idle

The Bus-Idle time may be of arbitrary length (minimum 0 bit). The bus is recognized to be free and a CAN Controller having information to transmit may access the bus. The detection of a dominant bit level during Bus-Idle on the bus is interpreted as the Start-of-Frame.

Bus Organization

Bus organization is based on five basic rules described in the following paragraphs.

Bus Access

CAN Controllers only start transmission during the Bus-Idle state. All CAN Controllers synchronize on the leading edge of the Start-of-Frame (hard synchronization).

Arbitration

If two or more CAN Controllers simultaneously start transmitting, the bus access conflict is solved by a bit-wise arbitration process during transmission of the Arbitration Field.

During arbitration every transmitting CAN Controller compares its transmitted bit level with the monitored bus level. Any CAN Controller which transmits a recessive bit and monitors a dominant bus level immediately becomes the receiver of the higher-priority message on the bus without corrupting any information on the bus. Each message contains an unique Identifier and a RTR bit describing the type of data within the message. The Identifier together with the RTR bit implicitly define the message's bus access priority. During arbitration the most significant bit of the Identifier is transmitted first and the RTR bit last. The message with the lowest binary value of the Identifier and RTR bit has the highest priority. A Data Frame has higher priority than a Remote Frame due to its RTR bit having a dominant level.

For every Data Frame there is an unique transmitter. For reasons of compatibility with other CAN-bus controllers, use of the Identifier bit pattern ID = $1111111XXXX_b$ (X being bits of arbitrary level) is forbidden.

The number of available different Identifiers is 2032 ($2^{11} - 2^4$).

Coding / Decoding

The following bit fields are coded using the bit-stuffing technique:

- Start-of-Frame
- Arbitration Field
- Control Field
- Data Field
- CRC Sequence

When a transmitting CAN Controller detects five consecutive bits of identical polarity to be transmitted, a complementary (stuff) bit is inserted into the transmitted bit-stream.

When a receiving CAN Controller has monitored five consecutive bits with identical polarity in the received bit streams of the above described bit fields, it automatically deletes the next received (stuff) bit. The level of the deleted stuff bit has to be the complement of the previous bits; otherwise a Stuff Error will be detected and signaled.

The remaining bit fields or frames are of fixed form and are not coded or decoded by the method of bit-stuffing.

The bit-stream in a message is coded according to the Non-Return-to-Zero (NRZ) method, i.e., during a bit period, the bit level is held constant, either recessive or dominant.

Error Signaling

A CAN Controller which detects an error condition, transmits an Error Flag. Whenever a Bit Error, Stuff Error, Form Error or an Acknowledgement Error is detected, transmission of an Error Flag is started at the next bit. Whenever a CRC Error is detected, transmission of an Error Flag starts at the bit following the Acknowledge Delimiter, unless an Error Flag for another error condition has already started. An Error Flag violates the bit-stuffing or corrupts the fixed form bit fields. A violation of the bit-stuffing law affects any CAN Controller which detects the error condition. These devices will also transmit an Error Flag.

An error-passive CAN Controller which detects an error condition, transmits a Passive Error Flag. A Passive Error Flag is not able to interrupt a current message at different CAN Controllers but this type of Error Flag may be ignored (overwritten) by other CAN Controllers. After having detected an error condition, an error-passive CAN Controller will wait for six consecutive bits with identical polarity and when monitoring them, interpret them as an Error Flag.

After transmission of an Error Flag, each CAN Controller monitors the bus-line until it detects a transition from a dominant-to-recessive bit level. At this point in time, every CAN Controller has finished transmitting its Error Flag and all CAN Controllers start transmitting seven additional recessive bits.

The message format of a Data Frame or Remote Frame is defined in such a way that all detectable errors can be signaled within the message transmission time and therefore it is very simple for the CAN Controllers to associate an Error Frame to the corresponding message and to initiate retransmission of the corrupted message.

If a CAN Controller monitors any deviation of the fixed form of an Error Frame, it transmits a new Error Frame.

Overload Signaling
Some CAN Controllers (but not the one on-chip of the 8XC592) require to delay the transmission of the next Data Frame or Remote Frame by transmitting one or more Overload Frames. The transmission of an Overload Frame must start during the first bit of an expected Intermission Field. Transmission of Overload Frames which are reactions on a dominant bit during an expected Intermission Field, start one bit after this event.

Though the format of Overload Frame and Error Frame are identical, they are treated differently. Transmission of an Overload Frame during Intermission Field does not initiate the retransmission of any previous Data Frame or Remote Frame.

If a CAN Controller which transmitted an Overload Frame monitors any deviation of its fixed form, it transmits an Error Frame.

Error Detection

The processes described in the following paragraphs are implemented in the 8XC592's on-chip CAN Controller for error detection.

Bit Error
A transmitting CAN Controller monitors the bus on a bit-by-bit basis. If the bit level monitored is different from the transmitted one, a Bit Error is signaled. The exceptions being:

- During the Arbitration Field, a recessive bit can be overwritten by a dominant bit. In this case, the CAN Controller interprets this as a loss of arbitration.
- During the Acknowledge Slot, only the receiving CAN Controllers are able to recognize a Bit Error.

Stuff Error
The following bit fields are coded using the bit-stuffing technique:

- Start-of-Frame
- Arbitration Field
- Control Field
- Data Field
- CRC Sequence

There are two possible ways of generating a Stuff Error:

- A disturbance generates more than the allowed five consecutive bits with identical polarity. These errors are detected by all CAN Controllers.
- A disturbance falsifies one or more of the five bits preceding the stuff bit. This error situation is not recognized as a Stuff Error by the receivers. Therefore, other error detection processes may detect this error condition such as: CRC check, format violation at the receiving CAN Controllers or Bit Error detection by the transmitting CAN Controller.

CRC Error
To ensure the validity of a transmitted message all receivers perform a CRC check. Therefore, in addition to the (destuffed) information digits (Start-of-Frame up to Data Field), every message includes some control digits (CRC Sequence; generated by the transmitting CAN Controller of the respective message) used for error detection.

The code used by all CAN Controllers is a (shortened) BCH code, extended by a parity check and has the following attributes:

- 127 bits as maximum length of the code
- 112 bits as maximum number of information digits (maximum 83 bits are used by the CAN Controller)
- Length of the CRC Sequence amounts to 15 bits
- Hamming distance d=6.

As a result, (d–1) random errors are detectable (some exceptions exist).

The CRC Sequence is determined (calculated) by the following procedure.

1. The destuffed bit stream consisting of Start-of-Frame up to the Data Field (if present) is interpreted as polynomial with coefficients 0 or 1.
2. This polynomial is divided (modulo–2) by the following generator polynomial, which includes a parity check:

   $f(X) = (X^{14} + X^9 + X^8 + X^6 + X^5 + X^4 + X^2 + X + 1)(X + 1) = 1100010110011001_b$.
3. The remainder of this polynomial division is the CRC sequence.

Burst errors are detected up to a length of 15 [degree of f(X)]. Multiple errors (number of disturbed bits at least d=6) are not detected with a residual error probability of 2^{-15} ($\approx 3 \cdot 10^{-5}$) by CRC check only.

Form Error
Form Errors result from violations of the fixed form of the following bit fields:

- CRC Delimiter
- Acknowledge Delimiter
- End-of-Frame
- Error Delimiter
- Overload Delimiter

During the transmission of these bit fields an error condition is recognized if a dominant bit level instead of a recessive one is detected.

Acknowledgement Error
This is detected by a transmitter whenever it does not monitor a dominant bit during the Acknowledge Slot.

Error Detection by an Error Flag of another CAN Controller
The detection of an error is signaled by transmitting an Error Flag. An Active Error Flag causes a Stuff Error, a Bit Error or a Form Error at all other CAN Controllers.

Error Detection Capabilities
Errors which occur at all CAN Controllers (global errors) are 100% detected. For local errors, i.e., for errors occurring at some CAN Controllers only, the shortened BCH code, extended by a parity check, has the following error detection capabilities:

- Up to five single Bit Errors are 100% detected, even if they are distributed randomly within the code
- All single Bit Errors are detected if their total number (within the code) is odd
- The residual error probability of the CRC check amounts to $3 \cdot 10^{-5}$.

Error Confinement (definitions)

Bus-Off
A CAN Controller which has too many unsuccessful transmissions, relative to the number of successful transmissions, will enter the Bus-Off state. It remains in this state, neither receiving nor transmitting messages until the Reset Request bit is set LOW (absent) and both Error Counters set to 0.

Acknowledge (ACK)
A CAN Controller which has received a valid message correctly, indicates this to the transmitter by transmitting a dominant bit level on the bus during the Acknowledge Slot, independent of accepting or rejecting the message.

Error-Active
An error–active CAN Controller in its normal operating state is able to receive and to transmit normally and also to transmit an Active Error Flag.

Error-Passive
An error–passive CAN Controller may transmit or receive messages normally. In the case of a detected error condition it transmits a Passive Error Flag instead of an Active Error Flag.

Hence the influence on bus activities by an error-active CAN Controller (e.g., due to a malfunction) is reduced.

Suspend Transmission
After an error-passive CAN Controller has transmitted a message, it sends eight recessive bits after the Intermission Field and then checks for Bus-Idle. If during Suspend Transmission another CAN Controller starts transmitting a message the suspended CAN Controller will become the receiver of this message; otherwise being in Bus-Idle it may start to transmit a further message.

Start-Up
A CAN Controller which either was switched off or is in the Bus-Off state, must run a Start-Up routine in order to:

- Synchronize with other available CAN Controllers before starting to transmit. Synchronization is achieved, when 11 recessive bits, equivalent to Acknowledge Delimiter, End-of-Frame and Intermission Field, have been detected (Bus-Free).
- Wait for other CAN Controllers without passing into the Bus-Off state (due to a missing acknowledge), if there is no other CAN Controller currently available.

Aims of Error Confinement

Distinction of Short and Long Lasting Disturbances
The CPU must be informed when there are long-lasting disturbances and when bus activities have returned to normal operation.

During long-lasting disturbances, a CAN Controller enters the Bus-Off state and the CPU may use default values.

Minor disturbances of bus activities will not affect a CAN Controller. In particular, a CAN Controller does not enter the Bus-Off state or inform the CPU of a short-lasting bus disturbance.

Detection and Localization of Hardware Disturbance and Defects
The rules for error confinement are defined by the CAN protocol specification (and implemented in the 8XC592's on-chip CAN Controller), in such a way that the CAN Controller, being nearest to the error-locus, reacts with a high probability the quickest (i.e., becomes error-passive or Bus-Off). Hence errors can be localized and their influence on normal bus activities is minimized.

Error Confinement
All CAN Controllers contain a Transmit Error Counter and a Receive Error Counter, which registers errors during the transmission and the reception of messages, respectively.

If a message is transmitted or received correctly, the count is decreased. In the event of an error, the count is increased. The Error Counters have an non-proportional method of counting: an error causes a larger counter increase than a correctly transmitted/received message causes the count to decrease. Over a period of time this may result in an increase in error counts, even if there are fewer corrupted messages than uncorrupted ones. The level of the Error Counters reflect the relative frequency of disturbances. The ratio of increase/decrease depends on the acceptable ration of invalid/valid messages on the bus and is hardware implemented to eight.

If one of the Error Counters exceeds the Warning Limit of 96 error points, indicating a significant accumulation of error conditions, this is signaled by the CAN Controller (Error Status, Error Interrupt).

A CAN Controller operates in the error-active mode until it exceeds 127 error points on one of its Error Counters. At this point it will enter the error-passive state.

A transmit error which exceeds 255 error points results in the CAN Controller entering the Bus-Off state.

INTERRUPT SYSTEM

External events and the real-time-driven on-chip peripherals require service by the CPU asynchronously to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution a multiple-source, two-priority-level, nested interrupt system is provided. Interrupt response latency is from 2.25µs to 7.5µs when using a 16MHz crystal. The latency time depends on the sequence of instructions executed directly after an interrupt request. During a CAN-DMA transfer the interrupt system is disabled. The 8XC592 acknowledges interrupt requests from fifteen sources as follows:

- $\overline{INT0}$ and $\overline{INT1}$: externally via pins 27 and 28 respectively
- Timer 0 and Timer 1: from the two internal counters
- Timer T2 (8 separate interrupts): 4 capture interrupts, 3 compare interrupts and an overflow interrupt
- ADC end-of-conversion interrupt
- CAN Controller interrupt
- UART serial I/O port interrupt.

Interrupt Enable Registers

IEN0 (A8H)

7	6	5	4	3	2	1	0
EA	EAD	ES1	ES0	ET1	EX1	ET0	EX0

Bit	Symbol	Function
IEN0.7	EA	General enable/disable control 0 = No interrupt is enabled 1 = Any individually enabled interrupt will be accepted
IEN0.6	EAD	Enable ADC interrupt
IEN0.5	ES1	Enable SIO1 (CAN) interrupt
IEN0.4	ES0	Enable SIO0 (UART) interrupt
IEN0.3	ET1	Enable Timer 1 interrupt
IEN0.2	EX1	Enable External 1 interrupt
IEN0.1	ET0	Enable Timer 0 interrupt
IEN0.0	EX0	Enable External 0 interrupt

IEN1 (E8H)

7	6	5	4	3	2	1	0
ET2	ECM2	ECM1	ECM0	ECT3	ECT2	ECT1	ECT0

Bit	Symbol	Function
IEN1.7	ET2	Enable T2 overflow interrupt(s)
IEN1.6	ECM2	Enable T2 comparator 2 interrupt
IEN1.5	ECM1	Enable T2 comparator 1 interrupt
IEN1.4	ECM0	Enable T2 comparator 0 interrupt
IEN1.3	ECT3	Enable T2 capture register 3 interrupt
IEN1.2	ECT2	Enable T2 capture register 2 interrupt
IEN1.1	ECT1	Enable T2 capture register 1 interrupt
IEN1.0	ECT0	Enable T2 capture register 0 interrupt

Where: 0 = interrupt disabled.
1 = interrupt enabled.

IP0 (B8H)

7	6	5	4	3	2	1	0
–	PAD	PS1	PS0	PT1	PX1	PT0	PX0

Bit	Symbol	Function
IP0.7	–	Unused
IP0.6	PAD	ADC interrupt priority level
IP0.5	PS1	SIO1 (CAN) interrupt priority level
IP0.4	PS0	SIO0 (UART) interrupt priority level
IP0.3	PT1	Timer 1 interrupt priority level
IP0.2	PX1	External interrupt 1 priority level
IP0.1	PT0	Timer 0 interrupt priority level
IP0.0	PX0	External interrupt 0 priority level

IP1 (F8H)

7	6	5	4	3	2	1	0
PT2	PCM2	PCM1	PCM0	PCT3	PCT2	PCT1	PCT0

Bit	Symbol	Function
IP1.7	PT2	T2 overflow interrupt(s) priority level
IP1.6	PCM2	T2 comparator 2 priority interrupt level
IP1.5	PCM1	T2 comparator 1 priority interrupt level
IP1.4	PCM0	T2 comparator 0 priority interrupt level
IP1.3	PCT3	T2 capture register 3 priority interrupt level
IP1.2	PCT2	T2 capture register 2 priority interrupt level
IP1.1	PCT1	T2 capture register 1 priority interrupt level
IP1.0	PCT0	T2 capture register 0 priority interrupt level

Where: 0 = interrupt disabled.
1 = interrupt enabled.

Table 10 shows the interrupt vectors. The vector indicates the Program memory location where the appropriate interrupt service routine starts.

Table 10. Interrupt Vectors

SOURCE		VECTOR
External 0	X0	0003H
Timer 0 overflow	T0	000BH
External 1	X1	0013H
Timer 1 overflow	T1	001BH
Serial I/O 0 (UART)	S0	0023H
Serial I/O 1 (CAN)	S1	002BH
T2 capture 0	CT0	0033H
T2 capture 1	CT1	003BH
T2 capture 2	CT2	0043H
T2 capture 3	CT3	004BH
ADC completion	ADC	0053H
T2 compare 0	CM0	005BH
T2 compare 1	CM1	0063H
T2 compare 2	CM2	006BH
T2 overflow	T2	0073H

Interrupt Priority

Each interrupt source can be iether high priority or low priority. If both priorities are requested simultaneously, the processor will branch to the highy priority vector. If there are simultaneous requests from sources of the same priority, then interrupts will be serviced in the following order: X0, S1, ADC, T0, CT0, CM0, X1, CT1, CM1, T1, CT2, CM2, S0, CT3, T2.

A low priority interrupt routine can only be interrupted by a high priority interrupt. A high priority interrupt routine cannot be interupted.

Table 11. Status of External Pins During Sleep, Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	PSEN	PORT 0	PORT 1	PORT 2	PORT 3	PORT 4	PWM0/ PWM1
Idle	Internal	1	1	Port Data	Port Data	Port Data	Port Data	Port Data	High
Idle	External	1	1	Float	Port Data	Address	Port Data	Port Data	High
Power-down	Internal	0	0	Port Data	Port Data	Port Data	Port Data	Port Data	High
Power-down	External	0	0	Float	Port Data	Port Data	Port Data	Port Data	High

NOTE:
If the port pins P1.6 and P1.7 are used as the CAN transmitter outputs (CTX0 and CTX1), then during Sleep and Power-down mode these pins output a "recessive" level.

POWER REDUCTION MODES

The 8XC592 has three software-selectable modes to reduce power consumption. These are:

- Sleep mode, affecting the CAN Controller only
- Idle mode, affecting the
  - CPU (halted)
  - Timer 2 (stopped and reset)
  - PWM0, PWM1 (reset, output = HIGH)
  - ADC (aborted if in progress)
- Power-down mode, affecting the whole 8XC592 device.

CAN Sleep Mode

In order to reduce power consumption of the P8XC592 the CAN Controller may be switched off (disconnecting the internal clock) by setting the CAN Command Register bit 4 (Sleep) HIGH. The CAN Controller leaves this Sleep mode by detecting either activity on the CAN-bus (dominant bit-level on CRX0/CXR1) or by setting the Sleep bit to LOW.

As the CPU cannot only write to the Sleep bit, but can also read it, the CAN Controller status can be determined directly.

Idle Mode

The instruction that sets PCON.0 is the last one executed in the normal operating mode before Idle mode is activated. Once in the Idle mode, the CPU status is preserved in its entirety: the Stack Pointer, Program Counter, Program Status Word, Accumulator, RAM and all other registers maintain their data during Idle mode. The status of the external pins during Idle mode is show in Table 12.

There are three ways to terminate the Idle mode:

Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, provided that the interrupt source is active during Idle mode. After the interrupt is serviced, the program continues with the instruction immediately after the one, at which the interrupt request was detected.

The flag bits GF0 and GF1 may be used to determine whether the interrupt was received during normal execution or during the Idle mode. For example, the instruction that writes to PCON.0 can also set or clear one or both flag bits. When Idle mode is terminated by an interrupt, the service routine can examine the status of the flag bits.

Another way of terminating the Idle mode is an external hardware reset. Since the oscillator is still running, the reset signal is required to be active only for 2 machine cycles (24 oscillator periods) to complete the reset operation.

The third way is the internally generated watchdog reset after an overflow of Timer 3.

Power-down Mode

The instruction that sets PCON.1 to HIGH, is the last one executed before entering the Power-down mode. In Power-down mode the oscillator of the P8XC592 is stopped. If the CAN Controller is in use, it is recommended to set it into Sleep mode before entering Power-down mode. However, setting PCON.1 to HIGH also sets the Sleep bit (CAN Controller Command Register bit 4) to HIGH.

The P8XC592 leaves Power-down mode either by a hardware reset or by a CAN Wake-up interrupt (due to activity on the CAN bus), if the SI01 (CAN) interrupt source is enabled (contents of register IEN0 = 1x1xxxxxb). A hardware reset affects the whole P8XC592, but leaves the contents of the on-chip RAM unchanged (CAN Controller and CPU's Special Function Registers are reset). A CAN Wake-up interrupt during Power-down mode causes a reset output pulse with a width of 6144 machine cycles (4.6ms with f_{CLK}=16MHz). All hardware except from the CAN Controller of the P8XC592 is reset (the contents of all CAN Controller registers are preserved).

Note that a capacitance connected to the RST pin could lengthen the internally generated reset pulse. If the pulse exceeds 8192 machine cycles, the CAN Controller part is reset too.

RESET

A reset is accomplished by holding the RST pin HIGH for at least two machine cycles (24 oscillator periods). The CPU responds by executing an internal reset. During reset ALE and PSEN output at a HIGH level. In order to perform a correct reset, this level must not be affected by external elements.

Also with the P8XC592, the RST line can be pulled HIGH internally by a pull-up transistor activated by the watchdog timer T3. The length of the output pulse from T3 is 3 machine cycles. A pulse of such short duration is necessary in order to recover from a processor or system fault as fast as possible.

During Power-down a reset could be generated internally via the CAN Wake-up interrupt. Then the RST pin is pulled HIGH for 6144 machine cycles. In this case the CAN Controller is not reset.

If the watchdog timer or the CAN Wake-up interrupt is used to reset external devices, the usual capacitor arrangement for power-on reset should not be used. However, the internal reset is forced, independent of the external level on the RST pin.

The Main RAM and AuxRAM are not affected. When VDD is turned on, the RAM content is indeterminate.

Table 12.
After a reset the internal registers have the following contents:

(CPU PART)	REGISTER	7	6	5	4	3	2	1	0
	ACC	0	0	0	0	0	0	0	0
	ADCO	X	X	0	0	0	0	0	0
	ADCH	X	X	X	X	X	X	X	X
	B	0	0	0	0	0	0	0	0
	CML0 – CML2	0	0	0	0	0	0	0	0
	CMH0 – CMH2	0	0	0	0	0	0	0	0
	CTCON	0	0	0	0	0	0	0	0
	CTL0 – CTL3	X	X	X	X	X	X	X	X
	CTH0 – CTH3	X	X	X	X	X	X	X	X
	DPL	0	0	0	0	0	0	0	0
	DPH	0	0	0	0	0	0	0	0
	IEN0	0	0	0	0	0	0	0	0
	IEN1	0	0	0	0	0	0	0	0
	IP0	X	0	0	0	0	0	0	0
	IP1	0	0	0	0	0	0	0	0
	PCH	0	0	0	0	0	0	0	0
	PCL	0	0	0	0	0	0	0	0
	PCON	0	X	X	0	0	0	0	0
	PSW	0	0	0	0	0	0	0	0
	PWM0	0	0	0	0	0	0	0	0
	PWM1	0	0	0	0	0	0	0	0
	PWMP	0	0	0	0	0	0	0	0
	P0 – P4	1	1	1	1	1	1	1	1
	P5	X	X	X	X	X	X	X	X
	RTE	0	0	0	0	0	0	0	0
	SOBUF	X	X	X	X	X	X	X	X
	SOCON	0	0	0	0	0	0	0	0
	CANSTA	0	0	0	0	1	1	0	0
	CANCON	X	X	X	0	0	0	0	0
	CANDAT	X	X	X	X	X	X	X	X
	CANADR	0	X	1	0	0	1	0	0
	SP	0	0	0	0	0	1	1	1
	STE	1	1	0	0	0	0	0	0
	TCON	0	0	0	0	0	0	0	0
	TH0, TH1	0	0	0	0	0	0	0	0
	TMH2	0	0	0	0	0	0	0	0
	TL0, TL1	0	0	0	0	0	0	0	0
	TML2	0	0	0	0	0	0	0	0
	TMOD	0	0	0	0	0	0	0	0
	TM2CON	0	0	0	0	0	0	0	0
	TM2IR	0	0	0	0	0	0	0	0
	T3	0	0	0	0	0	0	0	0
(CAN PART)	**REGISTER**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	CR	0	X	1	X	X	X	X	1
	CMR	1	1	X	0	X	X	X	X
	SR	0	0	0	0	1	1	0	0
	IR	X	X	X	0	0	0	0	0
	ACR	X	X	X	X	X	X	X	X
	AMR	X	X	X	X	X	X	X	X
	BTR 0	X	X	X	X	X	X	X	X
	BTR 1	X	X	X	X	X	X	X	X
	OCR	X	X	X	X	X	X	X	X
	TR	X	X	X	X	X	X	X	X
	TXB 10 – 19	X	X	X	X	X	X	X	X
	RXB 20 – 29	X	X	X	X	X	X	X	X

NOTE:
X = Undefined State

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING		UNIT
	MIN	MAX	
Storage temperature range	–65	+150	°C
Voltage on $\overline{EA}/V_{PP}$ to V_{SS}	–0.5	+13	V
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.0		W
Voltage on V_{DD} pin	–0.5	+6.5	V
Input voltage on any pin except from CTX0, CTX1, CRX0 and CRX1	–0.5	V_{DD}+0.5	V
Input output current on any I/O pin except fro CTX0 and CTX1	–	10	mA
Sink current of CTX0, CTX1 together	–	30	mA
Source current of CTX0, CTX1 together	–	20	mA

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_{amb} = –40°C to +85°C (83C592FFA, 87C592EFA), T_{amb} = –40°C to +125°C (83C592FHA); V_{DD}, AV_{DD} = 5V ±10%, V_{SS}, AV_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS TYPICAL	LIMITS MAX	UNIT
V_{DD}	Supply voltage		4.5		5.5	V
	Supply current:					
I_{DD}	operating	f_{CLK} = 16MHz	–		50	mA
I_{ID}	Idle mode	f_{CLK} = 16MHz	–		15	mA
I_{IS}	Idle and Sleep mode	f_{CLK} = 16MHz	–		10	mA
I_{PD}	Power-down mode (83C592 FHA)	Note 4			150	µA
I_{PD}	Power-down mode (8XC592 xFx)	Note 4			100	µA
Inputs						
V_{IL}	Input low voltage, except $\overline{EA}$, CRX0, CRX1		–0.5		$0.2V_{DD}$–0.1	V
V_{IL1}	Input low voltage to $\overline{EA}$		–0.5		$0.2V_{DD}$–0.3	V
V_{IH}	Input high voltage, except XTAL1, RST, CRX0, CRX1		$0.2V_{DD}$+0.9		V_{DD}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST		$0.7V_{DD}$		V_{DD}+0.5	V
$-I_{IL}$	Input current Low, ports 1, 2, 3, 4	V_{IN} = 0.45V			–50	µA
$-I_{TL}$	High-to-Low transition current, ports 1, 2, 3, 4	V_I = 2.0V/0.45V			–650	µA
$\pm I_{IL1}$	Input leakage current, port 0, $\overline{EA}$, STADC, $\overline{EW}$	0.45V<V_I<V_{DD}			10	µA
$\pm I_{IL2}$	Input leakage current, port 5	0.45V<V_I<V_{DD}			1	µA
Outputs						
V_{OL}	Output low voltage, ports 1, 2, 3, 4, except P1.6, P1.7[5, 6]	I_{OL} = 1.6mA[2]			0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$, $\overline{PWM0}$, $\overline{PWM1}$, P1.6, P1.7[5, 6]	I_{OL} = 3.2mA[2]			0.45	V
V_{OH}	Output high voltage, ports 1, 2, 3, 4	$-I_{OH}$ = 60µA $-I_{OH}$ = 25µA $-I_{OH}$ = 10µA	2.4 $0.75V_{DD}$ $0.9V_{DD}$			V V V
V_{OH1}	Output high voltage (port 0 in external bus mode, ALE, $\overline{PSEN}$, $\overline{PWM0}$, $\overline{PWM1}$)[3]	$-I_{OH}$ = 400µA $-I_{OH}$ = 150µA $-I_{OH}$ = 40µA	2.4 $0.75V_{DD}$ $0.9V_{DD}$			V V V
V_{OH2}	High level output voltage (RST)	$-I_{OH}$ = 400µA $-I_{OH}$ = 120µA	2.4 $0.8V_{DD}$			V
R_{RST}	Internal reset pull-down resistor		50		150	kΩ
C_{IO}	I/O buffer pin capacitance	Test freq = 1MHz, T_{amb} = 25°C			10	pF

DC ELECTRICAL CHARACTERISTICS (Continued)

T_{amb} = −40°C to +85°C (83C592FFA, 87C592EFA), T_{amb} = −40°C to +125°C (83C592FHA); V_{DD}, AV_{DD} = 5V ±10%, V_{SS}, AV_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS TYPICAL	LIMITS MAX	UNIT
Analog Inputs						
AV_{DD}	Analog supply voltage[7]	$AV_{DD} = V_{DD}$±0.2V	4.5		5.5	V
AI_{DD}	Analog supply current: Operating Idle mode	Port 5 = AV_{DD}			 2.5 2.5	 mA mA
AI_{IS}	Idle and sleep mode 83C592FHA 8XC592xFx				400 350	µA µA
AI_{PD}	Power-down mode 83C592FHA 8XC592xFx				400 350	µA µA
AV_{IN}	Analog input voltage		AV_{SS}−0.2		AV_{DD}+0.2	V
AV_{REF}	Reference voltage: AV_{REF-} AV_{REF+}		 AV_{SS}−0.2		 AV_{DD}+0.2	 V V
R_{REF}	Resistance between AV_{REF+} and AV_{REF-}		10		50	kΩ
C_{IA}	Analog input capacitance				15	pF
t_{ADS}	Sampling time				$8t_{CY}$	µs
t_{ADC}	Conversion time (including sampling time)				$50t_{CY}$	µs
DL_e	Differential non-linearity[8, 9, 10]				±1	LSB
IL_e	Integral non-linearity[8, 11]				±2	LSB
OS_e	Offset error[8, 12]				±2	LSB
G_e	Gain error[8, 13]				±0.4	%
A_e	Absolute Voltage Error[8, 14]				±3	LSB
M_{CTC}	Channel to channel matching				±1	LSB
C_t	Crosstalk between Port 5 inputs[9]	0–100kHz			−60	dB
CAN						
	CAN input comparator (CRX0, CRX1)	AV_{DD} = 5V ± 5% 1.4V<V_I<AV_{DD}−1.4V				
$±V_{DIF}$	Differential input voltage[15]		32		−	mV
V_{HYST}	Hysteresis voltage[15]		8		30	mV
$±I_I$	Input current		−		400	nA
	CAN output driver	V_{DD} = 5V ± 5%				
V_{OLT}	CTX0, CTX1, output voltage LOW	I_O = 1.2mA[15] I_O = 10mA	− −		0.1 0.6	V V
V_{OHT}	CTX0, CTX1, output voltage HIGH	I_O = 1.2mA[15] I_O = −10mA[16]	V_{DD} − 0.1 V_{DD} − 0.6		− −	V V
Reference						
	Reference	AV_{DD} = 5V ± 5%				
V_{REFOUT}	REF output voltage[15] (bit Reference Active = HIGH)	−0.1mA<I_L<0.1mA; C_L = 10nF	AV_{DD}/2−0.05		AV_{DD}/2+0.05	V
$±I_{REFIN}$	REF input current (bit Reference Active = LOW)	1.5V<V_{REFIN}< AV_{DD}−1.5V	−		10	µA

NOTES: SEE NEXT PAGE.

NOTES TO THE DC ECLECTRICAL CHARACTERISTICS:

1. The operating current is measured with all output pins unloaded; XTAL1 is driven with $t_R = t_F = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; $\overline{EA}$ = RST = Port 0 = P1.6 = P1.7 = $\overline{EW} = V_{DD}$; STADC = V_{SS}; CRX0 = 2.7V; CRX1 = 2.3V.
2. The idle mode supply current is measured with all output pins unloaded; XTAL1 is driven with $t_R = t_F = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; Port 0 = P1.6 = P1.7 = $\overline{EW} = V_{DD}$; $\overline{EA}$ = RST = STADC = V_{SS}; CRX0 = 2.7V; CRX1 = 2.3V.
3. The idle and sleep mode supply current is measured with all output pins unloaded; XTAL1 is driven with $t_R = t_F = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{DD} - 0.5V$; Port 0 = P1.6 = P1.7 = $\overline{EW}$ = CRX0 = V_{DD}; $\overline{EA}$ = RST = STADC = CRX1 = V_{SS}; CAN: register 6: = 00H, register 7: = 12H, register 8: = 02H, register 0: = 20H, wait $15t_{CY}$, register 1: = 10H, wait for bit Sleep = 1.
4. The power-down current is measured with all output pins unloaded; Port 0 = P1.6 = P1.7 = $\overline{EW}$ = CRX0 = V_{DD}; $\overline{XTAL1}$ = $\overline{EA}$ = RST = STADC = CRX1 = V_{SS}. Windowed devices must have the window covered during testing.
5. Under steady state (non-transient) conditions, I_{OL} must be limited externally as follows:

   Maximum I_{OL} per 8 bit port
   - Port 0: 26mA
   - Port 1: 32mA
   - Ports 2, 3, and 4: 15mA

   Maximum I_{OL} for all output pins: 71mA

   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
6. Capacitive loads on Port0 and Port2 may degrade the LOW level output voltage of ALE, Port1 and Port2. During a 1-to-0 transition on the Port0 and Port2 pins and a capacitive load > 100pF, the ALE LOW level may exceed 0.8V. In that case, it is necessary to connect ALE to a Schmitt trigger input respectively use an address latch with a Schmitt trigger STROBE input.
7. Capacitive loads on Port0 and Port2 may cause a HIGH level output voltage degradation of ALE and $\overline{PSEN}$ below 0.9 V_{DD} during the address bits are stabilizing.
8. $AV_{REF+} = 5.12V$; $AV_{REF-} = 0V$; $AV_{DD} = 5.0V$.
9. The differential non-linearity (DL_e) is the difference between the actual step width and the ideal step width.
10. The ADC is monotonic, there are no missing codes.
11. The integral non-linearity (IL_e) is the peak difference between the center of the steps of the actual and the ideal transfer curve after appropriate adjustment of gain and offset error.
12. The offset error (OS_e) is the absolute difference between the straight line which fits the actual transfer curve after remiving gain error, and a straight line which fits the ideal transfer curve. The offset error is constant at every point of the actual transfer curve.
13. The gain error (G_e) is the relative difference in percent between the straight line fitting the actual transfer curve after removing offset error and the straight line which fits the ideal transfer curve. The gain error is constant at every point on the transfer curve.
14. The absolute voltage error (A_e) is the maximum difference between the center of the steps of the actual transfer curve of the not calibrated ADC and the ideal transfer curve.
15. Not tested during production.
16. Source current for the CTX0, CTX1 outputs together.

AC ELECTRICAL CHARACTERISTICS

T_{amb} = –40°C to +85°C, V_{DD}, AV_{DD} = 5V ±10%, V_{SS}, AV_{SS} = 0V

SYMBOL	FIGURE	PARAMETER	VARIABLE CLOCK		UNIT
			MIN	MAX	
$1/t_{CLK}$	7	Oscillator frequency	1.2	16	MHz
t_{LHLL}	7	ALE pulse width	$2t_{CLK}$–40		ns
t_{AVLL}	7	Address valid to ALE low	t_{CLK}–55		ns
t_{LLAX}	7	Address hold after ALE low	t_{CLK}–35		ns
t_{LLIV}	7	ALE low to valid instruction in		$4t_{CLK}$–100	ns
t_{LLPL}	7	ALE low to $\overline{PSEN}$ low	t_{CLK}–40		ns
t_{PLPH}	7	$\overline{PSEN}$ pulse width	$3t_{CLK}$–45		ns
t_{PLIV}	7	$\overline{PSEN}$ low to valid instruction in		$3t_{CLK}$–105	ns
t_{PXIX}	7	Input instruction hold after $\overline{PSEN}$	0		ns
t_{PXIZ}	7	Input instruction float after $\overline{PSEN}$		t_{CLK}–25	ns
t_{AVIV}	7	Address to valid instruction in		$5t_{CLK}$–105	ns
t_{PLAZ}	7	$\overline{PSEN}$ low to address float		10	ns
Data Memory					
t_{AVLL}	8, 9	Address valid to ALE low	t_{CLK}–55		ns
t_{RLRH}	8, 9	$\overline{RD}$ pulse width	$6t_{CLK}$–100		ns
t_{WLWH}	8, 9	$\overline{WR}$ pulse width	$6t_{CLK}$–100		ns
t_{RLDV}	8, 9	$\overline{RD}$ low to valid data in		$5t_{CLK}$–165	ns
t_{RHDX}	8, 9	Data hold after $\overline{RD}$	0		ns
t_{RHDZ}	8, 9	Data float after $\overline{RD}$		$2t_{CLK}$–70	ns
t_{LLDV}	8, 9	ALE low to valid data in		$8t_{CLK}$–150	ns
t_{AVDV}	8, 9	Address to valid data in		$9t_{CLK}$–165	ns
t_{LLWL}	8, 9	ALE low to $\overline{RD}$ or $\overline{WR}$ low	$3t_{CLK}$–50	$3t_{CLK}$+50	ns
t_{AVWL}	8, 9	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	$4t_{CLK}$–130		ns
t_{QVWX}	8, 9	Data valid to $\overline{WR}$ transition	t_{CLK}–60		ns
t_{WHQX}	8, 9	Data hold after $\overline{WR}$	t_{CLK}–50		ns
t_{RLAZ}	8, 9	$\overline{RD}$ low to address float		0	ns
t_{WHLH}	8, 9	$\overline{RD}$ or $\overline{WR}$ high to ALE high	t_{CLK}–40	t_{CLK}+40	ns
External Clock					
t_{CHCX}	11	High time[3]	20		ns
t_{CLCX}	11	Low time[3]	20		ns
t_{CLCH}	11	Rise time[3]		20	ns
t_{CHCL}	11	Fall time[3]		20	ns
UART Timing in Shift Register Mode					
t_{XLXL}	10	Serial port clock cycle time[3]	$12t_{CLK}$		µs
t_{QVXH}	10	Output data setup to clock rising edge	$10t_{CLK}$–133		ns
t_{XHQX}	10	Output data hold after clock rising edge	$2t_{CLK}$–117		ns
t_{XHDX}	10	Input data hold after clock rising edge	0		ns
t_{XHDV}	10	Clock rising edge to input data valid		$10t_{CLK}$–133	ns
CAN Input Comparator / Output Driver (AV_{DD} = 5V ± 5%)					
t_{SD}		Sum of input and output delay (V_{DIF} = ± 32mV; 1.4V < V_I < AV_{DD} – 1.4V) AV_{DD} = 5V ± 5%	–	60	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. These values are characterized but not 100% production tested.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. the designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruciton (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

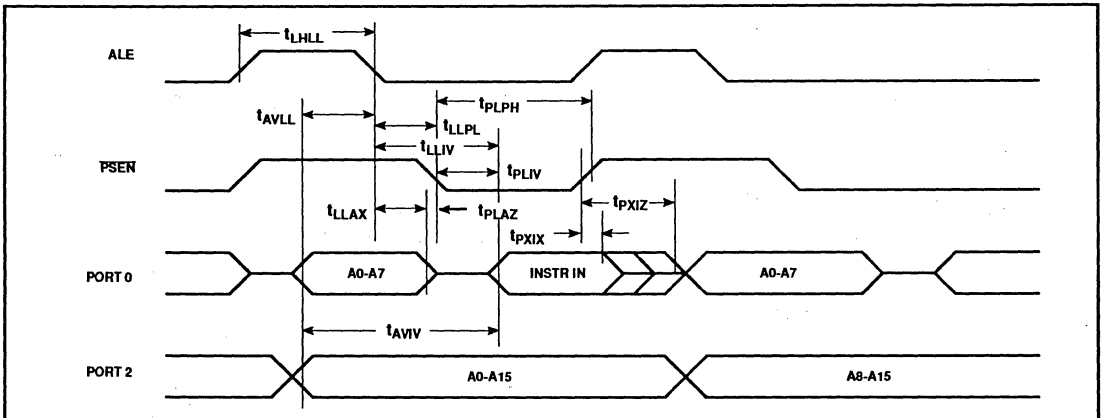


Figure 7. External Program Memory Read Cycle

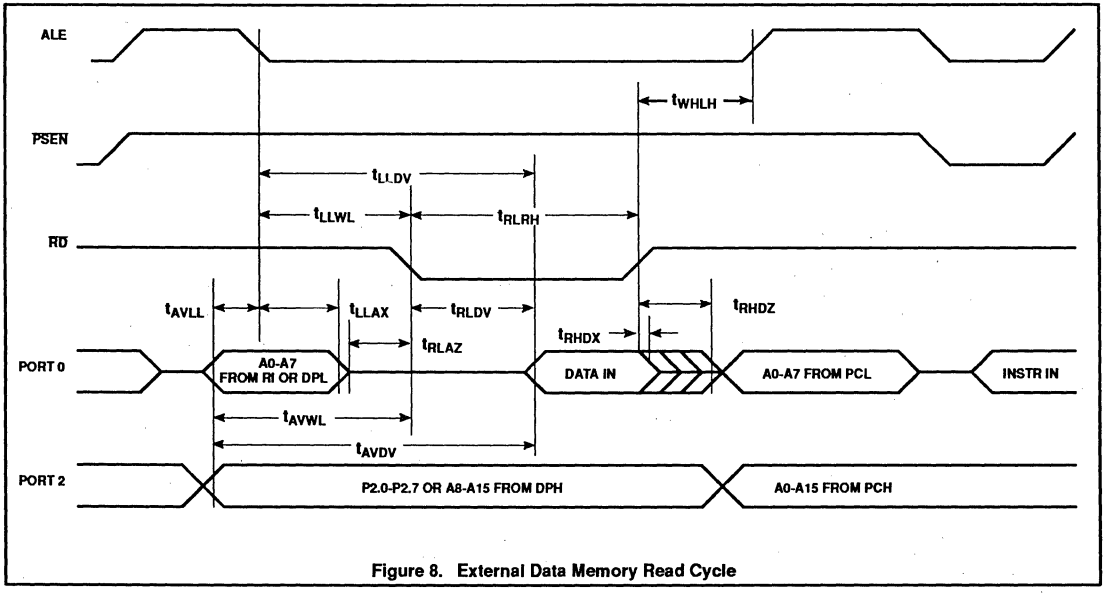


Figure 8. External Data Memory Read Cycle

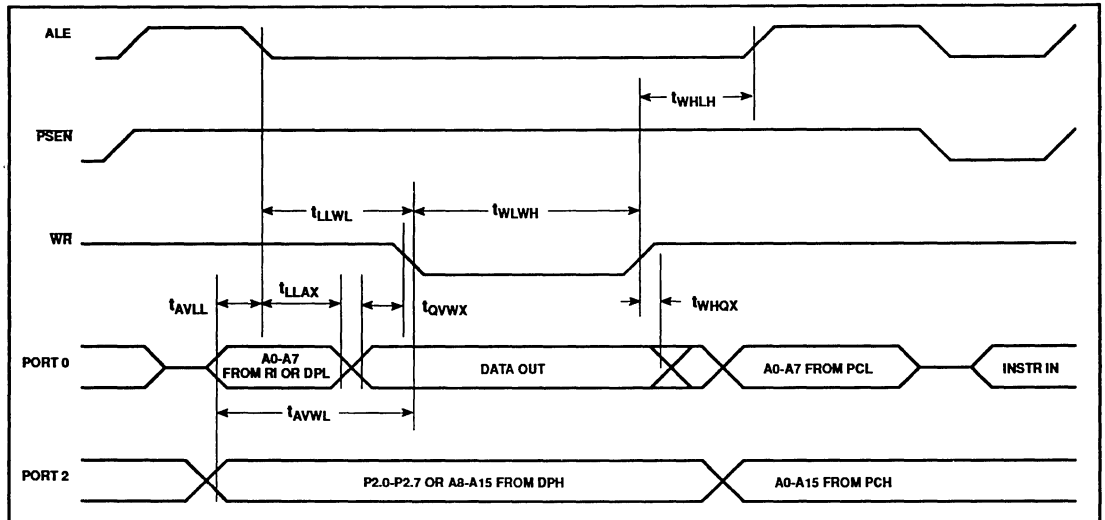


Figure 9. External Data Memory Write Cycle

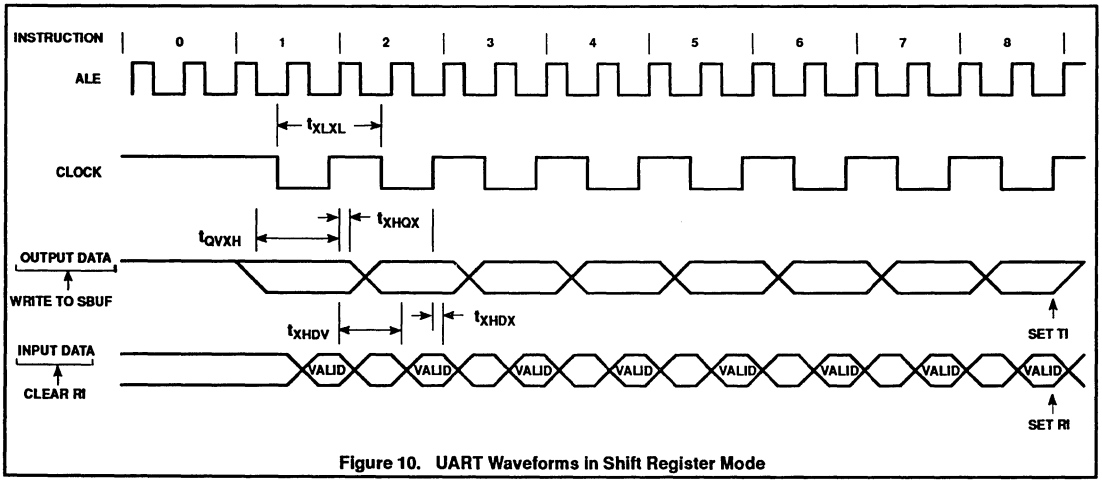


Figure 10. UART Waveforms in Shift Register Mode

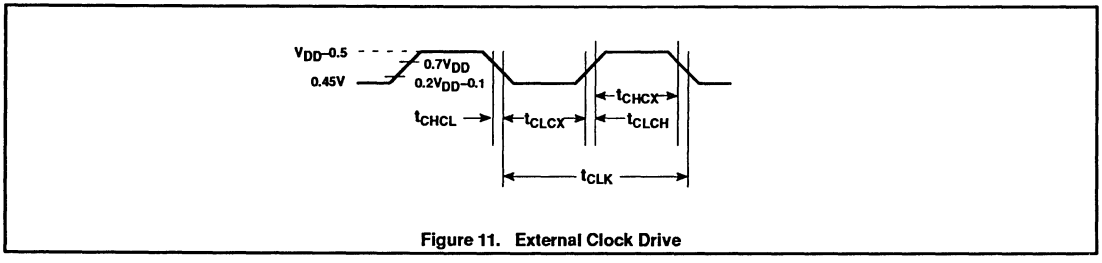
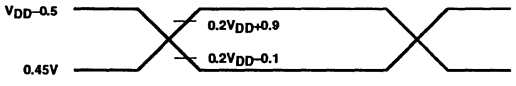
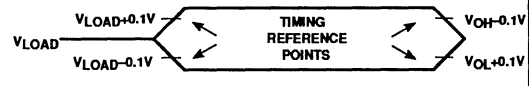


Figure 11. External Clock Drive



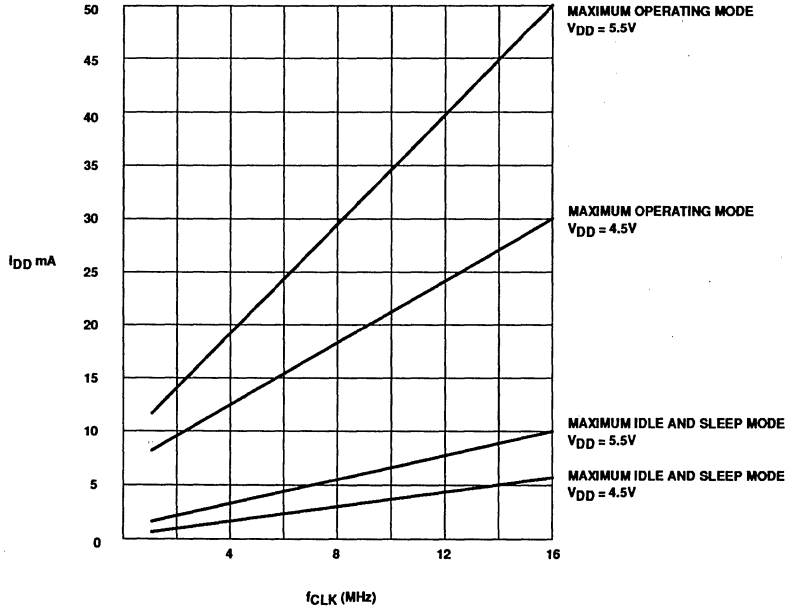
NOTE:
AC inputs during testing are driven at V_{DD} –0.5 for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at V_{IH} min for a logic '1' and V_{IL} max for a logic '0'.

Figure 12. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 13. Float Waveform



NOTE:
Valid only within frequency specifications of the device under test. Maximum values taken at worst case temperature.

Figure 14. I_{DD} vs. FREQ

EPROM CHARACTERISTICS

The 87C592 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C592 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C592 manufactured by Signetics.

Table 13 shows the logic levels for reading the signature byte, and for programming the program memory, theencryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 15 and 16. Figure 17 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 15. Note that the 87C592 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 15. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$, and pins of ports 2 and 3 specified in Table 13 are held at the "Program Code Data" levels indicated in Table 13. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 16.

To program the encryption table, repeat the 25-pulse programming sequence for addresses 0 through 1FH, using the "Pgm Encryption Table" levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25-pulse programming sequence using the "Pgm Lock Bit" levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be red is applied to ports 1 and 2 as shown in Figure 17. The other pins are held at the "Verify Code Data" levels indicated in Table 13. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(031H) = 9CH indicates 87C592

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 13, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to the light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345-5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000µW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient. Erasure leaves the array in an all 1s state.

Table 13. EPROM Programming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signagure	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:
1. 0 = Valid low for that pin; 1 = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{DD} = 5V ±10% during programming and verification.

* ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

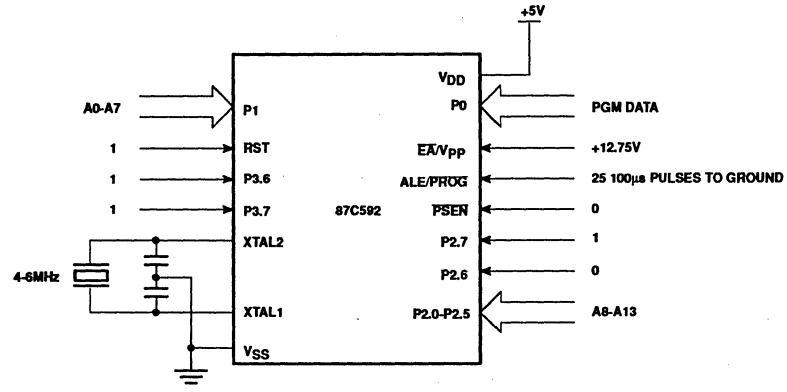


Figure 15. Programming Configuration

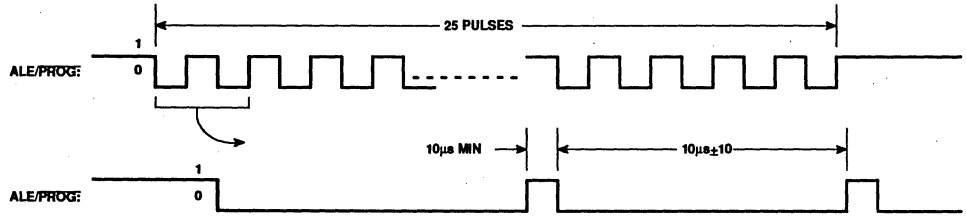


Figure 16. PROG Waveform

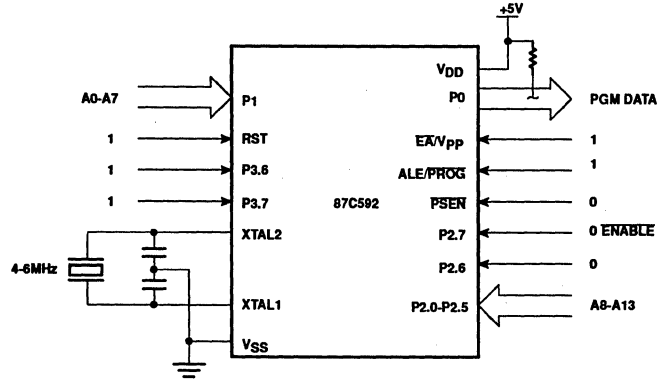


Figure 17. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_{amb} = 21°C to +27°C, V_{DD} = 5V±10%, V_{SS} = 0V (See Figure 18)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLK}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLK}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLK}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLK}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLK}$		
t_{EHSH}	P2.7 (ENABLE) high to V_{PP}	$48t_{CLK}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLK}$	
t_{ELQV}	$\overline{ENABLE}$ low to data valid		$48t_{CLK}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLK}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

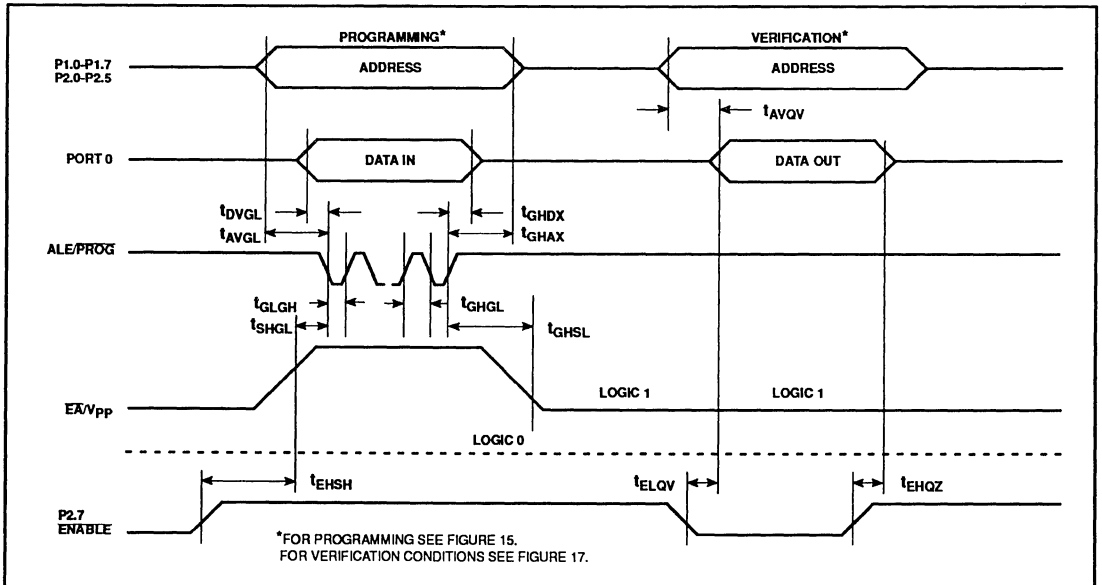


Figure 18. EPROM Programming and Verification

8XC652/654 OVERVIEW

The 8XC652 and 8XC654 (hereafter referred to collectively as 8XC652/4) are derivatives of the 80C51 8-bit CMOS microcontroller. The 8XC652/4 contains all of the features of the 80C51 (that is, the standard counter/timers T0 and T1, the standard serial I/O (UART), and four 8-bit I/O ports). In addition, the 8XC652/4 has the following:

- 8k bytes of ROM (8XC652)
- 16k bytes of ROM (8XC654)
- 256 bytes of RAM
- I^2C bus serial I/O

The only difference between the 8XC652 and the 8XC654 is that the 8XC654 has 16k bytes of program memory while the 8XC652 has 8k bytes. All other features of these parts are identical.

The 8XC652/4 is pin-for-pin compatible and fully code compatible with the 80C51. There are some differences in the P1.6 and P1.7 pin functions that are described in detail later in this section. All of the 80C51 functions are present, including the external 64k program and data memory expansion, Boolean processing, and two reduced power modes.

Differences from the 80C51

The data and program memory are organized similar to the 80C51. The 8XC652/4 program memory differs in that it has 8k/16k bytes of on-chip ROM. When $\overline{EA}$ is high the 8XC652/4 fetches instructions from the internal ROM unless the address exceeds 1FFFH/3FFFH. Locations 2000H/4000H to FFFFH are fetched from external program memory. When $\overline{EA}$ is held low, all instruction fetches are from external memory.

The organization of the data memory is similar to the 80C51 except that the 8XC652/4 has an additional 128 bytes of RAM overlapped with the special function register space. This additional RAM is addressed using indirect addressing only and is available as stack space. (This memory addition is the same as in the 80C52 and 83C552. See Figure 1 for a memory map.)

Special Function Registers

The 8XC652/4 special function register space is the same as that on the 80C51 except that it contains four additional SFRs. The added registers are: S1CON, S1STA, S1DAT,, and S1ADR. In addition to these, the standard UART special function registers SCON and SBUF have been renamed S0CON and S0BUF for clarity.

Since the standard 80C51 on-chip functions are the same on the 8XC652/4, the SFR locations, bit locations, and operation are unchanged. The only exception is in the interrupt enable and interrupt priority SFRs. These have been changed to include the interrupt from the I^2C serial port. Table 1 shows the special function registers, their direct address, the bit addresses, and the value in the register after a reset.

I^2C Serial Communication—SIO1

The I^2C serial port is identical to the I^2C serial port on the 8XC552. The operation of this subsystem is described in detail in the 8XC552 section of this manual.

Note that in both the 8XC652/4 and the 8XC552 the I^2C pins are alternate functions to port pins P1.6 and P1.7. Because of this, P1.6 and P1.7 on these parts do not have a pull-up structure as found on the 80C51. Therefore P1.6 and P1.7 have open drain outputs on the 8XC652/4.

Idle and Power-Down Operation

Idle mode operation permits the interrupt, serial ports, and timer blocks to continue to function while the CPU is halted. The following functions remain active during idle mode. These functions may generate an interrupt or reset and thus end the idle mode:

- Timer 1 overflow
- I^2C serial I/O interrupt
- UART serial I/O interrupt
- Timer 0, Timer 1
- SIO0, SIO1
- External interrupt

In idle mode, port pins P1.6 and P1.7 function as SCL and SDA, respectively, if the I^2C serial port is enabled. The power-down operation freezes the oscillator. The power-down mode can only be activated by setting the PD bit in the PCON register. The power-down mode in the 8XC652/4 operates exactly the same as in the 80C51.

ROM Code Protection (83C652/83C654)

The 83C652/83C654 has an additional security feature. ROM code protection is mask programmable and therefore user dependent. This feature may be requested during ROM code submission. When enabled, access to the internal ROM is only possible when executing from internal program memory, not in the EA-mode (external access).

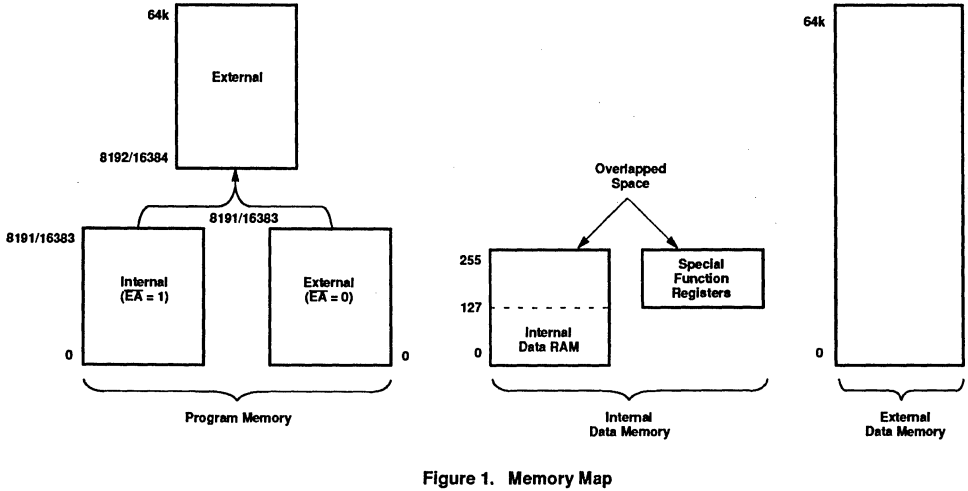


Figure 1. Memory Map

Table 1. 8XC652/654 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*#	Interrupt enable	A8H	EA		ES1	ES0	ET1	EX1	ET0	EX0	0x000000B
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*#	Interrupt priority	B8H	–		PS1	PS0	PT1	PX1	PT0	PX0	xx000000B
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
			97	96	95	94	93	92	91	90	
P1*#	Port 1	90H	SDA	SCL							FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	A15	A14	A13	A12	A11	A10	A9	A8	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	$\overline{RD}$	$\overline{WR}$	T1	T0	$\overline{INT1}$	$\overline{INT0}$	TXD	RXD	FFH
PCON	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxx0000B
			9F	9E	9D	9C	9B	9A	99	98	
S0CON*#	Serial 0 port control	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
S0BUF#	Serial 0 data buffer	99H									xxxxxxxxB
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	F1	P	00H
S1DAT#	Serial 1 data	DAH									00H
SP	Stack pointer	81H									07H
S1ADR#	Serial 1 address	DBH	SLAVE ADDRESS							GC	00H
S1STA#	Serial 1 status	D9H	SC4	SC3	SC2	SC1	SC0	0	0	0	F8H
			DF	DE	DD	DC	DB	DA	D9	D8	
S1CON*#	Serial 1 control	D8H	CR2	ENS1	STA	STO	SI	AA	CR1	CR0	00000000B
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
TH1	Timer high 1	8DH									00H
TH0	Timer high 0	8CH									00H
TL1	Timer low 1	8BH									00H
TH0	Timer low 0	8AH									00H
TMOD	Timer mode	89H	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	00H

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.

Interrupt System

The interrupt system is the same as in the 80C51 except that the 8XC652/4 acknowledges interrupt requests from six sources as follows:

- $\overline{INT0}$ external interrupt 0
- $\overline{INT1}$ external interrupt 1
- Timer 0 overflow
- Timer 1 overflow
- I^2C serial I/O interrupt
- UART serial interrupt

See Figure 2 for a function diagram of the 8XC652/4 interrupt structure. Each interrupt vectors to a separate location in program memory for its service program. Each source can be individually enabled or disabled by a corresponding bit in the IE register; moreover, each interrupt may be programmed to a high or low priority level using a corresponding bit in the IP register. Also, all enabled sources can be globally disabled or enabled.

Both external interrupts can be programmed to be level-activated or transition-activated; an active LOW level allows "Wire-ORing" or several input sources to the input pin.

Each interrupt source can be set for either high priority or low priority. If two separate interrupts are requested simultaneously, the processor will branch to the vector associated with the interrupt that has the higher priority. If there are simultaneous requests from sources that have the same priority, then the interrupts will be serviced in the following order:

1. $\overline{INT0}$ external interrupt 0
2. I^2C serial I/O interrupt
3. Timer 0 overflow
4. $\overline{INT1}$ external interrupt 1
5. Timer 1 overflow
6. UART serial I/O interrupt

A low priority interrupt routine can be interrupted by an interrupt having a higher priority. A high priority interrupt cannot be interrupted. All of the features of the 8XC652/4 that have not been discussed in this section are the same as those on the 80C51.

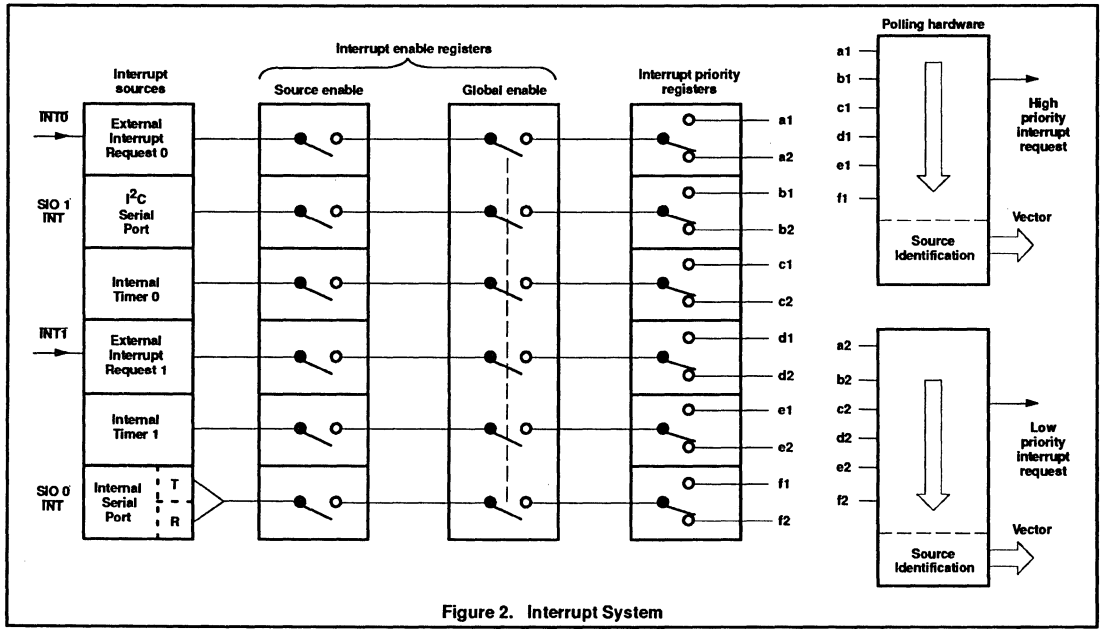
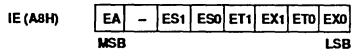


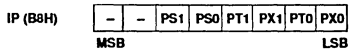
Figure 2. Interrupt System

Interrupt Enable Register



Bit	Symbol	Function
IE.7	EA	General enable/disable control 0 = No interrupt enabled 1 = Any individually enabled interrupt will be accepted
IE.6	–	Unused
IE.5	ES1	Enable SIO1 (I^2C) interrupt
IE.4	ES0	Enable SIO0 (UART) interrupt
IE.3	ET1	Enable timer 1 interrupt
IE.2	EX1	Enable external 1 interrupt
IE.1	ET0	Enable timer 0 interrupt
IE.0	EX0	Enable external 0 interrupt 0 = interrupt disabled 1 = interrupt enabled

Interrupt Priority Register



Bit	Symbol	Function
IP.7	–	Unused
IP.6	–	Unused
IP.5	PS1	SIO1 (I^2C) interrupt priority level
IP.4	PS0	SIO0 (UART) interrupt priority level
IP.3	PT1	Timer 1 interrupt priority level
IP.2	PX1	Enable interrupt 1 priority level
IP.1	PT0	Timer 0 interrupt priority level
IP.0	PX0	External interrupt 0 priority level 0 = Low priority 1 = High priority

The following vectors indicate the ROM location where the appropriate interrupt service routine starts.

Source	Vector
External 0 (EX0)	0003H
Timer 0 overflow (T0)	000BH
External 1 (EX1)	0013H
Timer 1 overflow (T1)	001BH
Serial I/O 0 (UART) (S0)	0023H
Serial I/O 1 (I^2C) (S1)	002BH

CMOS single-chip 8-bit microcontroller — 80C652/83C652/87C652

DESCRIPTION

The 80C652/83C652/87C652 Single-Chip 8-Bit Microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 80C652/83C652/87C652 has the same instruction set as the 80C51. Three versions of the derivative exist:

83C652 — 8k bytes mask programmable ROM

80C652 — ROMless version

87C652 — EPROM version

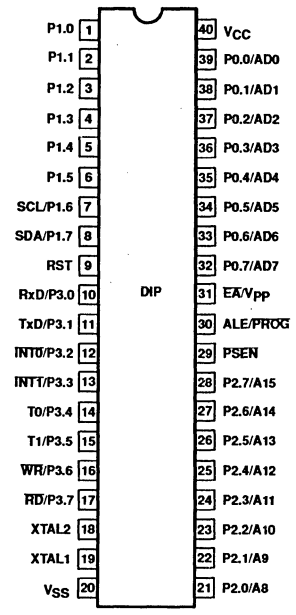
This device provides architectural enhancements that make it applicable in a variety of applications for general control systems. The 8XC652 contains a non-volatile 8k × 8 read-only program mem-ory (83C652) EPROM (87C652), a volatile 256 × 8 read/write data memory, four 8-bit I/O ports, two 16-bit timer/event counters (identical to the timers of the 80C51), a multi-source, two-priority-level, nested interrupt structure, an I²C interface, UART and on-chip oscillator and timing circuits. For systems that require extra capability, the 8XC652 can be expanded using standard TTL compatible memories and logic.

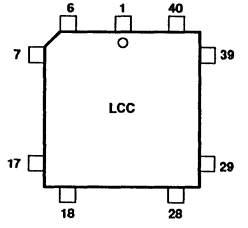
The device also functions as an arithmetic processor having facilities for both binary and BCD arithmetic plus bit-handling capabilities. The instruction set consists of over 100 instructions: 49 one-byte, 45 two-byte and 17 three-byte. With a 16MHz crystal, 58% of the instructions are executed in 0.75µs and 40% in 1.5µs. Multiply and divide instructions require 3µs.

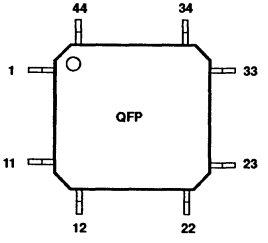
FEATURES

- 80C51 central processing unit
- 8k × 8 ROM expandable externally to 64k bytes (87C652 EPROM is not expandable)
- 256 × 8 RAM, expandable externally to 64k bytes
- Two standard 16-bit timer/counters
- Four 8-bit I/O ports
- I²C-bus serial I/O port with byte oriented master and slave functions
- Full-duplex UART facilities
- Power control modes
  - Idle mode
  - Power-down mode
- ROM code protection
- Five package styles
- Extended temperature ranges
- OTP package available
- Three speed ranges
  - 16MHz
  - 20MHz (87C652 only)
  - 24MHz (80C652/83C652 only)

PIN CONFIGURATION

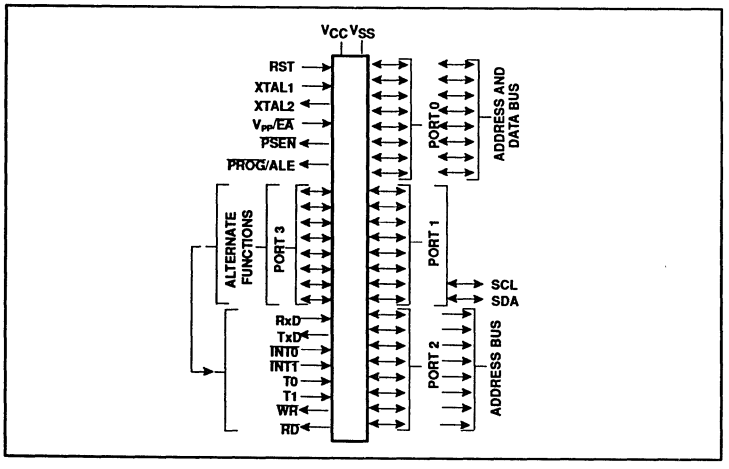






SEE PAGE 524 FOR QFP AND LCC PIN FUNCTIONS.

LOGIC SYMBOL

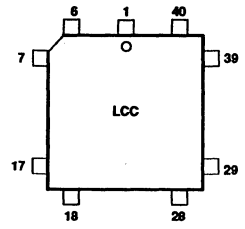


PART NUMBER SELECTION

PHILIPS PART ORDER NUMBER PART MARKING		SIGNETICS PART ORDER NUMBER			TEMPERATURE (°C)	
ROMless	ROM[4]	ROMless	ROM	EPROM	AND PACKAGE	FREQUENCY
P80C652FBP	P83C652FBP/xxx	S80C652-4N40	S83C652-4N40	S87C652-4N40	0 to +70, plastic DIP	16MHz
				S87C652-4F40	0 to +70, ceramic DIP with window	16MHz
P80C652FBA	P83C652FBA/xxx	S80C652-4A44	S83C652-4A44	S87C652-4A44	0 to +70, plastic PLCC	16MHz
				S87C652-4K44	0 to +70, ceramic CLCC with window	16MHz
P80C652FBB	P83C652FBB/xxx	S80C652-4B44	S83C652-4B44	S87C652-4B44	0 to +70, plastic QFP	16MHz
P80C652FFP	P83C652FFP/xxx	S80C652-5N40	S83C652-5N40	S87C652-5N40	–40 to +85, plastic DIP	16MHz
				S87C652-5F40	–40 to +85, ceramic DIP with window	16MHz
P80C652FFA	P83C652FFA/xxx	S80C652-5A44	S83C652-5A44	S87C652-5A44	–40 to +85, plastic PLCC	16MHz
				S87C652-5K44	–40 to +85, ceramic CLCC with window	16MHz
F80C652FFB	P83C652FFB/xxx	S80C652-5B44	S83C652-5B44	S87C652-5B44	–40 to +85, plastic QFP	16MHz
P80C652FHP	P83C652FHP/xxx	S80C652-6N40	S83C652-6N40		–40 to +125, plastic DIP	16MHz
P80C652FHA	P83C652FHA/xxx	S80C652-6A44	S83C652-6A44		–40 to +125, plastic PLCC	16MHz
P80C652FHB	P83C652FHB/xxx	S80C652-6B44	S83C652-6B44		–40 to +125, plastic QFP	16MHz
				S87C652-7N40	0 to +70, plastic DIP	20MHz
				S87C652-7F40	0 to +70, ceramic DIP with window	20MHz
				S87C652-7A44	0 to +70, plastic PLCC	20MHz
				S87C652-7K44	0 to +70, ceramic CLCC with window	20MHz
				S87C652-7B44	0 to +70, plastic QFP	20MHz
				S87C652-8N40	–40 to +85, plastic DIP	20MHz
				S87C652-8F40	–40 to +85, ceramic DIP with window	20MHz
				S87C652-8A44	–40 to +85, plastic PLCC	20MHz
				S87C652-8K44	–40 to +85, ceramic CLCC with window	20MHz
				S87C652-8B44	–40 to +85, plastic QFP	20MHz
P80C652IBP	P83C652IBP/xxx	S80C652-AN40	S83C652-AN40		0 to +70, plastic DIP	24MHz
P80C652IBA	P83C652IBA/xxx	S80C652-AA44	S83C652-AA44		0 to +70, plastic PLCC	24MHz
P80C652IBB	P83C652IBB/xxx	S80C652-AB44	S83C652-AB44		0 to +70, plastic QFP	24MHz
P80C652IFP	P83C652IFP/xxx	S80C652-BN40	S83C652-BN40		–40 to +85, plastic DIP	24MHz
P80C652IFA	P83C652IFA/xxx	S80C652-BA44	S83C652-BA44		–40 to +85, plastic PLCC	24MHz
P80C652IFB	P83C652IFB/xxx	S80C652-BB44	S83C652-BB44		–40 to +85, plastic QFP	24MHz

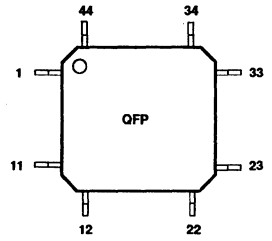
NOTES:
1. 80C652 and 83C652 frequency range is 1.2MHz–16MHz.
2. 87C652 frequency range is 3.5MHz–16MHz or 3.5MHz–20MHz.
3. The 87C652 EPROM is not expandable.
4. xxx denotes the ROM code number.

LCC PIN FUNCTIONS



Pin	Function	Pin	Function
1	NC	23	NC
2	P1.0	24	P2.0/A8
3	P1.1	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6/SCL	30	P2.6/A14
9	P1.7/SDA	31	P2.7/A15
10	RST	32	$\overline{PSEN}$
11	P3.0/RxD	33	ALE/$\overline{PROG}$
12	NC	34	NC
13	P3.1/TxD	35	$\overline{EA}$/V_{PP}
14	P3.2/$\overline{INT0}$	36	P0.7/AD7
15	P3.3/$\overline{INT1}$	37	P0.6/AD6
16	P3.4/T0	38	P0.5/AD5
17	P3.5/T1	39	P0.4/AD4
18	P3.6/$\overline{WR}$	40	P0.3/AD3
19	P3.7/$\overline{RD}$	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	V_{SS}	44	V_{CC}

QFP PIN FUNCTIONS

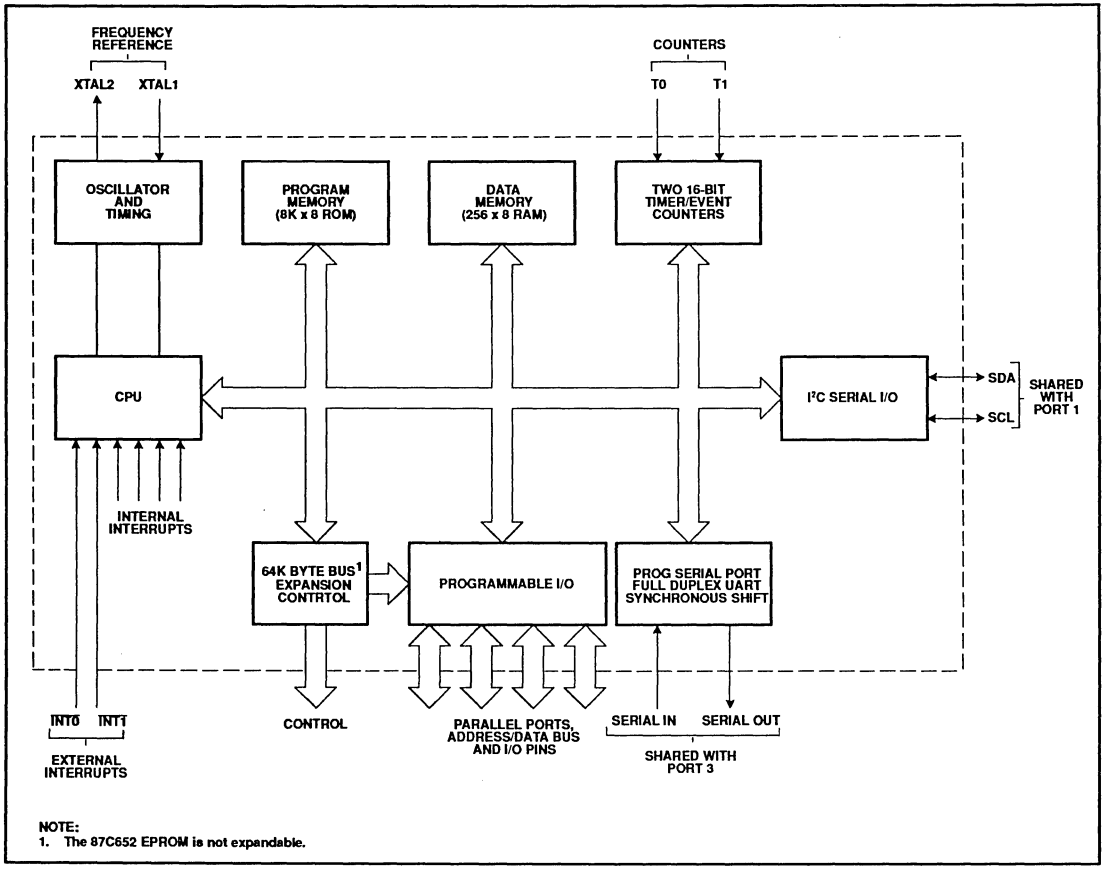


Pin	Function	Pin	Function
1	P1.5	23	P2.5/A13
2	P1.6/SCL	24	P2.6/A14
3	P1.7/SDA	25	P2.7/A15
4	RST	26	$\overline{PSEN}$
5	P3.0/RxD	27	ALE/$\overline{PROG}$
6	NC	28	NC
7	P3.1/TxD	29	$\overline{EA}$/V_{PP}
8	P3.2/$\overline{INT0}$	30	P0.7/AD7
9	P3.3/$\overline{INT1}$	31	P0.6/AD6
10	P3.4/T0	32	P0.5/AD5
11	P3.5/T1	33	P0.4/AD4
12	P3.6/$\overline{WR}$	34	P0.3/AD3
13	P3.7/$\overline{RD}$	35	P0.2/AD2
14	XTAL2	36	P0.1/AD1
15	XTAL1	37	P0.0/AD0
16	V_{SS}	38	V_{CC}
17	NC	39	NC
18	P2.0/A8	40	P1.0
19	P2.1/A9	41	P1.1
20	P2.2/A10	42	P1.2
21	P2.3/A11	43	P1.3
22	P2.4/A12	44	P1.4

NOTES TO QFP ONLY:

1. Due to EMC improvements, it is advised to connect pins 6, 28, 39 to V_{SS} on the 80C652/83C652.

BLOCK DIAGRAM



NOTE:
1. The 87C652 EPROM is not expandable.

PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{SS}	20	22	16	I	**Ground:** 0V reference.
V_{CC}	40	44	38	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–36	37–30	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 also outputs the code bytes during program verification in the 87C652. External pull-ups are required during program verification.
P1.0–P1.7	1–8	2–9	40–44, 1–3	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups, except P1.6 and P1.7 which are open drain. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 1 also receives the low-order address byte during program memory verification. Alternate functions include:
P1.6	7	8	2	I/O	**SCL:** I^2C-bus serial port clock line.
P1.7	8	9	3	I/O	**SDA:** I^2C-bus serial port data line.
P2.0–P2.7	21–28	24–31	18–25	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	5, 7–13	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 80C51 family, as listed below:
	10	11	5	I	**RxD (P3.0):** Serial input port
	11	13	7	O	**TxD (P3.1):** Serial output port
	12	14	8	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	9	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	10	I	**T0 (P3.4):** Timer 0 external input
	15	17	11	I	**T1 (P3.5):** Timer 1 external input
	16	18	12	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	13	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	4	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE/$\overline{PROG}$	30	33	27	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{PSEN}$	29	32	26	O	**Program Store Enable:** The read strobe to external program memory. When the 87C652 is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$/V_{PP}	31	35	29	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H and 1FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 1FFFH. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	19	21	15	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	14	O	**Crystal 2:** Output from the inverting oscillator amplifier.

NOTE:
To avoid "latch-up" effect at power-on, the voltage on any pin at any time must not be higher than V_{CC} + 0.5V or V_{SS} – 0.5V, respectively.

ROM CODE PROTECTION (83C652)

The 83C652 has an additional security feature. ROM code protection may be selected by setting a mask programmable security bit (i.e., user dependent). This feature may be requested during ROM code submission. When selected, the ROM code is protected and cannot be read out at any time by any test mode or by any instruction in the external program memory space.

The MOVC instructions are the only instructions that have access to program code in the internal or external program memory. The $\overline{EA}$ input is latched during RESET and is "don't care" after RESET. This implementation prevents reading internal program code by switching from external program memory to internal program memory during a MOVC instruction or any other instruction that uses immediate data.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 522.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-on reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

IDLE MODE

In the idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON. Table 1 shows the state of the I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Mode

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Serial Control Register (S1CON) – See Table 2

S1CON (D8H)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

Bits CR0, CR1 and CR2 determine the serial clock frequency that is generated in the master mode of operation.

Table 2. Serial Clock Rates

CR2	CR1	CR0	BIT FREQUENCY (kHz) AT f_{osc} 6MHz	12MHz	16MHz	24MHz	f_{osc} DIVIDED BY
0	0	0	23	47	62.5	94	256
0	0	1	27	54	71	107[1]	224
0	1	0	31.25	62.5	83.3	125[1]	192
0	1	1	37	75	100	150[1]	160
1	0	0	6.25	12.5	17	25	960
1	0	1	50	100	133[1]	200[1]	120
1	1	0	100	200[1]	267[1]	400[1]	60
1	1	1	> 0.25 < 62.5	> 0.5 < 62.5	> 0.67 < 56	> 0.98 < 50	96 × (256 – (reload value Timer 1)) (Reload value range: 0 – 254 in mode 2)

NOTES:
1. These frequencies exceed the upper limit of 100kHz of the I²C-bus specification and cannot be used in an I²C-bus application.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ to V_{SS} (87C652 only)	–0.5 to + 13	V
Voltage on any other pin to V_{SS}	–0.5 to + 6.5	V
Input, output current on any single pin	±5	mA
Input, output current on any two pins	±10	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DEVICE SPECIFICATIONS

TYPE	SUPPLY VOLTAGE (V)		FREQUENCY (MHz)		TEMPERATURE
	MIN.	MAX.	MIN.	MAX.	(°C)
P83(0)C652FB	4.0	6.0	1.2	16	0 to +70
S87C652–4	4.5	5.5	3.5	16	0 to +70
P83(0)C652FF	4.0	6.0	1.2	16	–40 to +85
S87C652–5	4.5	5.5	3.5	16	–40 to +85
P83(0)C652FH	4.5	5.5	1.2	16	–40 to +125
S87C652–7	4.5	5.5	3.5	20	0 to +70
S87C652–8	4.5	5.5	3.5	20	–40 to +85
P83(0)C652IB	4.5	5.5	1.2	24	0 to +70
P83(0)C652IF	4.5	5.5	1.2	24	–40 to +85

DC ELECTRICAL CHARACTERISTICS

V_{SS} = 0V

SYMBOL	PARAMETER	PART TYPE	TEST CONDITIONS	LIMITS MIN.	LIMITS MAX.	UNIT
V_{IL}	Input low voltage, except $\overline{EA}$, P1.6/SCL, P1.7/SDA	0°C to +70°C –40°C to +85°C –40°C to +125°C		–0.5 –0.5 –0.5	$0.2V_{CC} - 0.1$ $0.2V_{CC} - 0.15$ $0.2V_{CC} - 0.25$	V V V
V_{IL1}	Input low voltage to $\overline{EA}$	0°C to +70°C –40°C to +85°C –40°C to +125°C		–0.5 –0.5 –0.5	$0.2V_{CC} - 0.3$ $0.2V_{CC} - 0.35$ $0.2V_{CC} - 0.45$	V V V
V_{IL2}	Input low voltage to P1.6/SCL, P1.7/SDA[6]			–0.5	1.5	V
V_{IH}	Input high voltage, except XTAL1, RST, P1.6/SCL, P1.7/SDA	0°C to +70°C –40°C to +85°C –40°C to +125°C		$0.2V_{CC} + 0.9$ $0.2V_{CC} + 1.0$ $0.2V_{CC} + 1.0$	$V_{CC} + 0.5$ $V_{CC} + 0.5$ $V_{CC} + 0.5$	V V V
V_{IH1}	Input high voltage, XTAL1, RST	0°C to +70°C –40°C to +85°C –40°C to +125°C		$0.7V_{CC}$ $0.7V_{CC} + 0.1$ $0.7V_{CC} + 0.1$	$V_{CC} + 0.5$ $V_{CC} + 0.5$ $V_{CC} + 0.5$	V V V
V_{IH2}	Input high voltage, P1.6/SCL, P1.7/SDA[6]			3.0	6.0	V
V_{OL}	Output low voltage, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA		I_{OL} = 1.6mA[8]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$		I_{OL} = 3.2mA[8]		0.45	V
V_{OL2}	Output low voltage, P1.6/SCL, P1.7/SDA		I_{OL} = 3.0mA		0.4	V
V_{OH}	Output high voltage, ports 1, 2, 3		I_{OH} = –60µA I_{OH} = –25µA I_{OH} = –10µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$		V V V
V_{OH1}	Output high voltage, Port 0 in external bus mode, ALE, $\overline{PSEN}$, RST[9]		I_{OH} = –400µA I_{OH} = –150µA I_{OH} = –40µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$		V V V
I_{IL}	Logical 0 input current, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA	0°C to +70°C –40°C to +85°C –40°C to +125°C	V_{IN} = 0.45V		–50 –75 –75	µA µA µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA	0°C to +70°C –40°C to +85°C –40°C to +125°C	See Note 7		–650 –750 –750	µA µA µA
I_{L1}	Input leakage current, port 0		$0.45 < V_I < V_{CC}$		±10	µA
I_{L2}	Input leakage current, P1.6/SCL, P1.7/SDA		$0V < V_I < 6.0V$ $0V < V_{CC} < 6.0V$		±10	µA µA
I_{CC}	Power supply current: Active mode @ 16MHz (80/83C652) [2, 10] Active mode @ 16MHz (87C652) [2] Idle mode @ 16MHz (80/83C652) [3, 10] Idle mode @ 16MHz (87C652) [3] Power down mode [4, 5] Power down mode [4, 5]	 –40°C to +125°C	See Note 1 V_{CC} = 6.0V		 26.5 25 6 6 50 100	 mA mA mA mA µA µA
R_{RST}	Internal reset pull-down resistor			50	150	kΩ
C_{IO}	Pin Capacitance		Freq. = 1MHz		10	pF

NOTES: See next page.

NOTES FOR DC ELECTRICAL CHARACTERISTICS:

1. See Figures 10 through 13 for I_{CC} test conditions.
2. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{CC} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = P1.6 = P1.7 = V_{CC}; f_{CLK} = 16MHz. See Figure 10.
3. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{CC} - 0.5V$; XTAL2 not connected; Port 0 = P1.6 = P1.7 = V_{CC}; $\overline{EA}$ = RST = V_{SS}; f_{CLK} = 16MHz. See Figure 11.
4. The power-down current is measured with all output pins disconnected; XTAL2 not connected; Port 0 = P1.6 = P1.7 = V_{CC}; $\overline{EA}$ = RST = V_{SS}. See Figure 13.
5. $2V \leq V_{PD} \leq V_{CC}$max.
6. The input threshold voltage of P1.6 and P1.7 (SIO1) meets the I^2C specification, so an input voltage below 1.5V will be recognized as a logic 0 while an input voltage above 3.0V will be recognized as a logic 1.
7. Pins of ports 1 , 2, and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
8. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. I_{OL} can exceed these conditions provided that no single output sinks more than 5mA and no more than two outputs exceed the test conditions.
9. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
10. I_{CCMAX} for the 80/83C652 at other frequencies can be derived from Figure 1, where FREQ is the external oscillator frequency in MHz. I_{CCMAX} is given in mA.

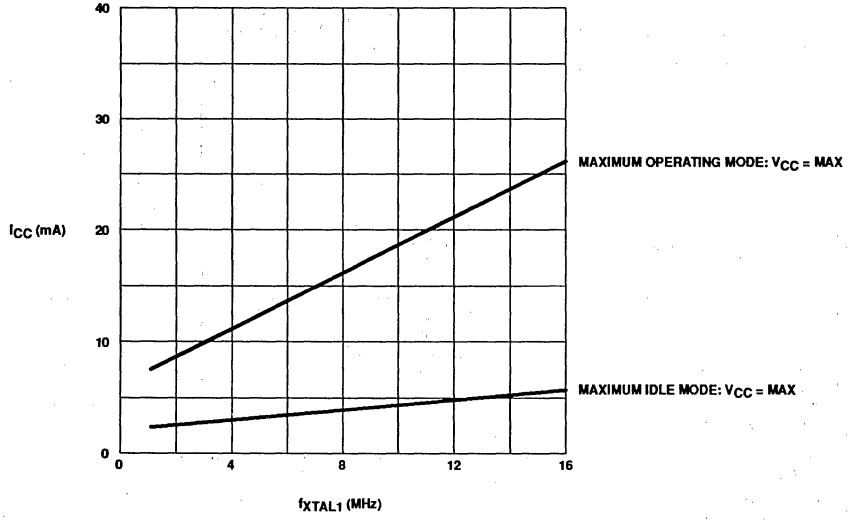


Figure 1. I_{CC} vs. Frequency

AC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C, or T_{amb} = –40°C to +85°C/+125°C, V_{SS} = 0V[1,2]

SYMBOL	FIGURE	PARAMETER	16MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	2	Oscillator frequency			1.2 (3.5)	16	MHz
t_{LHLL}	2	ALE pulse width	85		$2t_{CLCL}$–40		ns
t_{AVLL}	2	Address valid to ALE low	8		t_{CLCL}–55		ns
t_{LLAX}	2	Address hold after ALE low	28		t_{CLCL}–35		ns
t_{LLIV}	2	ALE low to valid instruction in		150		$4t_{CLCL}$–100	ns
t_{LLPL}	2	ALE low to $\overline{PSEN}$ low	23		t_{CLCL}–40		ns
t_{PLPH}	2	$\overline{PSEN}$ pulse width	143		$3t_{CLCL}$–45		ns
t_{PLIV}	2	$\overline{PSEN}$ low to valid instruction in		83		$3t_{CLCL}$–105	ns
t_{PXIX}	2	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	2	Input instruction float after $\overline{PSEN}$		38		t_{CLCL}–25	ns
t_{AVIV}	2	Address to valid instruction in		208		$5t_{CLCL}$–105	ns
t_{PLAZ}	2	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{AVLL}	3, 4	Address valid to ALE low	28		t_{CLCL}–35		ns
t_{RLRH}	3, 4	$\overline{RD}$ pulse width	275		$6t_{CLCL}$–100		ns
t_{WLWH}	3, 4	$\overline{WR}$ pulse width	275		$6t_{CLCL}$–100		ns
t_{RLDV}	3, 4	$\overline{RD}$ low to valid data in		148		$5t_{CLCL}$–165	ns
t_{RHDX}	3, 4	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	3, 4	Data float after $\overline{RD}$		55		$2t_{CLCL}$–70	ns
t_{LLDV}	3, 4	ALE low to valid data in		350		$8t_{CLCL}$–150	ns
t_{AVDV}	3, 4	Address to valid data in		398		$9t_{CLCL}$–165	ns
t_{LLWL}	3, 4	ALE low to $\overline{RD}$ or $\overline{WR}$ low	138	238	$3t_{CLCL}$–50	$3t_{CLCL}$+50	ns
t_{AVWL}	3, 4	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	120		$4t_{CLCL}$–130		ns
t_{QVWX}	3, 4	Data valid to $\overline{WR}$ transition	3		t_{CLCL}–60		ns
t_{DW}	3, 4	Data setup time before $\overline{WR}$	288		$7t_{CLCL}$–150		ns
t_{WHQX}	3, 4	Data hold after $\overline{WR}$	13		t_{CLCL}–50		ns
t_{RLAZ}	3, 4	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	3, 4	$\overline{RD}$ or $\overline{WR}$ high to ALE high	23	103	t_{CLCL}–40	t_{CLCL}+40	ns
Shift Register[3]							
t_{XLXL}	5	Serial port clock cycle time[4]	0.75		$12t_{CLCL}$		µs
t_{QVXH}	5	Output data setup to clock rising edge[4]	492		$10t_{CLCL}$–133		ns
t_{XHQX}	5	Output data hold after clock rising edge[4]	8		$2t_{CLCL}$–117		ns
t_{XHDX}	5	Input data hold after clock rising edge[4]	0		0		ns
t_{XHDV}	5	Clock rising edge to input data valid[4]		498		$10t_{CLCL}$–133	ns
External Clock							
t_{CHCX}	6	High time[4]	20		20	$t_{CLCL} - t_{LOW}$	ns
t_{CLCX}	6	Low time[4]	20		20	$t_{CLCL} - t_{HIGH}$	ns
t_{CLCH}	6	Rise time[4]		20		20	ns
t_{CHCL}	6	Fall time[4]		20		20	ns

AC ELECTRICAL CHARACTERISTICS (Continued)

T_{amb} = 0°C to +70°C, or T_{amb} = –40°C to +85°C/+125°C, V_{SS} = 0V[1, 2]

SYMBOL	FIGURE	PARAMETER	24MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	2	Oscillator frequency			1.2 (3.5)	24 (20)	MHz
t_{LHLL}	2	ALE pulse width	43		$2t_{CLCL}$–40		ns
t_{AVLL}	2	Address valid to ALE low	17		t_{CLCL}–25		ns
t_{LLAX}	2	Address hold after ALE low	17		t_{CLCL}–25		ns
t_{LLIV}	2	ALE low to valid instruction in		102		$4t_{CLCL}$–65	ns
t_{LLPL}	2	ALE low to $\overline{PSEN}$ low	17		t_{CLCL}–25		ns
t_{PLPH}	2	$\overline{PSEN}$ pulse width	80		$3t_{CLCL}$–45		ns
t_{PLIV}	2	$\overline{PSEN}$ low to valid instruction in		65		$3t_{CLCL}$–60	ns
t_{PXIX}	2	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	2	Input instruction float after $\overline{PSEN}$		17		t_{CLCL}–25	ns
t_{AVIV}	2	Address to valid instruction in		128		$5t_{CLCL}$–80	ns
t_{PLAZ}	2	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{AVLL}	3, 4	Address valid to ALE low	17		t_{CLCL}–25		ns
t_{RLRH}	3, 4	$\overline{RD}$ pulse width	150		$6t_{CLCL}$–100		ns
t_{WLWH}	3, 4	$\overline{WR}$ pulse width	150		$6t_{CLCL}$–100		ns
t_{RLDV}	3, 4	$\overline{RD}$ low to valid data in		118		$5t_{CLCL}$–90	ns
t_{RHDX}	3, 4	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	3, 4	Data float after $\overline{RD}$		55		$2t_{CLCL}$–28	ns
t_{LLDV}	3, 4	ALE low to valid data in		183		$8t_{CLCL}$–150	ns
t_{AVDV}	3, 4	Address to valid data in		210		$9t_{CLCL}$–165	ns
t_{LLWL}	3, 4	ALE low to $\overline{RD}$ or $\overline{WR}$ low	75	175	$3t_{CLCL}$–50	$3t_{CLCL}$+50	ns
t_{AVWL}	3, 4	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	92		$4t_{CLCL}$–75		ns
t_{QVWX}	3, 4	Data valid to $\overline{WR}$ transition	12		t_{CLCL}–30		ns
t_{DW}	3, 4	Data setup time before $\overline{WR}$	162		$7t_{CLCL}$–130		ns
t_{WHQX}	3, 4	Data hold after $\overline{WR}$	17		t_{CLCL}–25		ns
t_{RLAZ}	3, 4	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	3, 4	$\overline{RD}$ or $\overline{WR}$ high to ALE high	17	67	t_{CLCL}–25	t_{CLCL}+25	ns
Shift Register[3]							
t_{XLXL}	5	Serial port clock cycle time[4]	0.5		$12t_{CLCL}$		µs
t_{QVXH}	5	Output data setup to clock rising edge[4]	283		$10t_{CLCL}$–133		ns
t_{XHQX}	5	Output data hold after clock rising edge[4]	23		$2t_{CLCL}$–60		ns
t_{XHDX}	5	Input data hold after clock rising edge[4]	0		0		ns
t_{XHDV}	5	Clock rising edge to input data valid[4]		283		$10t_{CLCL}$–133	ns
External Clock							
t_{CHCX}	6	High time[4]	17		17	$t_{CLCL} - t_{LOW}$	ns
t_{CLCX}	6	Low time[4]	17		17	$t_{CLCL} - t_{HIGH}$	ns
t_{CLCH}	6	Rise time[4]		20		20	ns
t_{CHCL}	6	Fall time[4]		20		20	ns

AC ELECTRICAL CHARACTERISTICS (Continued)

T_{amb} = 0°C to +70°C, or T_{amb} = –40°C to +85°C/+125°C, V_{SS} = 0V[1, 2]

SYMBOL	PARAMETER	INPUT	OUTPUT
I²C Interface			
$t_{HD;STA}$	START condition hold time	≥ 14 t_{CLCL}	> 4.0µs [5]
t_{LOW}	SCL low time	≥ 16 t_{CLCL}	> 4.7µs [5]
t_{HIGH}	SCL high time	≥ 14 t_{CLCL}	> 4.0µs [5]
t_{RC}	SCL rise time	≤ 1µs	– [6]
t_{FC}	SCL fall time	≤ 0.3µs	< 0.3µs [7]
$t_{SU;DAT1}$	Data set-up time	≥ 250ns	> 20 t_{CLCL} – t_{RD}
$t_{SU;DAT2}$	SDA set-up time (before rep. START cond.)	≥ 250ns	> 1µs [5]
$t_{SU;DAT3}$	SDA set-up time (before STOP cond.)	≥ 250ns	> 8 t_{CLCL}
$t_{HD;DAT}$	Data hold time	≥ 0ns	> 8 t_{CLCL} – t_{FC}
$t_{SU;STA}$	Repeated START set-up time	≥ 14 t_{CLCL} [5]	> 4.7µs [5]
$t_{SU;STO}$	STOP condition set-up time	≥ 14 t_{CLCL} [5]	> 4.0µs [5]
t_{BUF}	Bus free time	≥ 14 t_{CLCL} [5]	> 4.7µs [5]
t_{RD}	SDA rise time	≤ 1µs [8]	– [6]
t_{FD}	SDA fall time	≤ 300ns [8]	< 0.3µs [7]

NOTES:
1. Parameters are valid over operating temperature range and voltage range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. The shift register has been characterized for the 87C652 only.
4. These values are characterized but not 100% production tested.
5. At 100 kbit/s. At other bit rates this value is inversely proportional to the bit-rate of 100 kbit/s.
6. Determined by the external bus-line capacitance and the external bus-line pull-resistor, this must be < 1µs.
7. Spikes on the SDA and SCL lines with a duration of less than 3 t_{CLCL} will be filtered out. Maximum capacitance on bus-lines SDA and SCL = 400pF.
8. $t_{CLCL} = 1/f_{OSC}$ = one oscillator clock period at pin XTAL1. For 63ns < t_{CLCL} < 285ns (16MHz > f_{OSC} > 3.5MHz) the I²C interface meets the I²C-bus specification for bit-rates up to 100 kbit/s.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

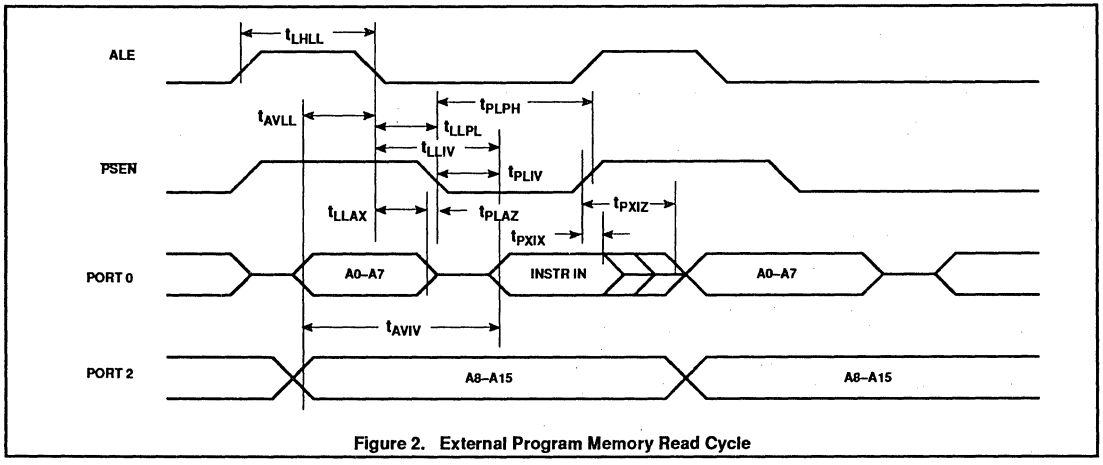


Figure 2. External Program Memory Read Cycle

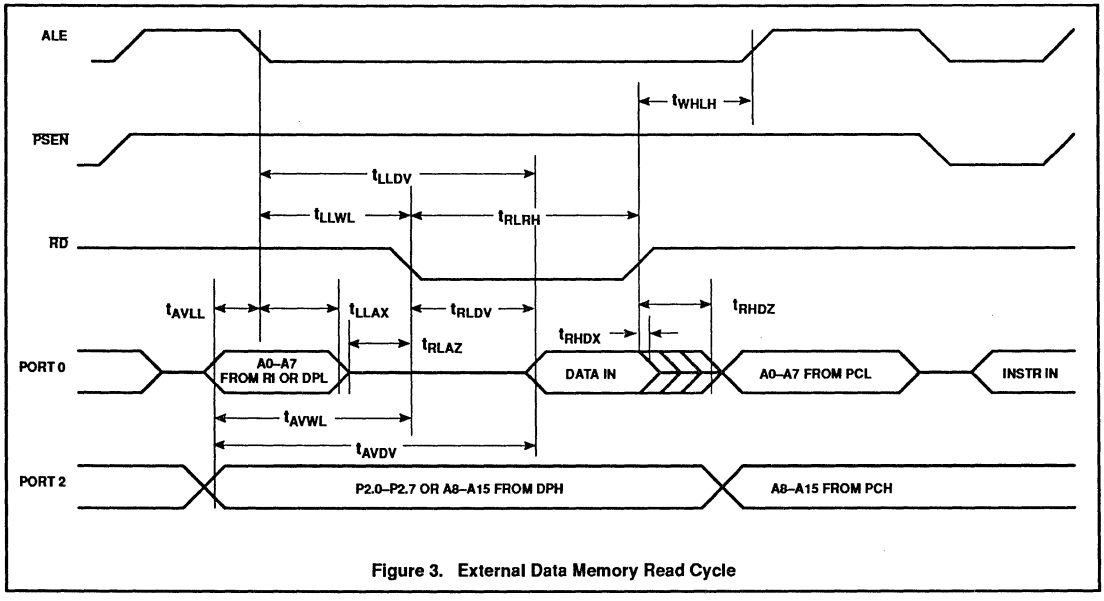


Figure 3. External Data Memory Read Cycle

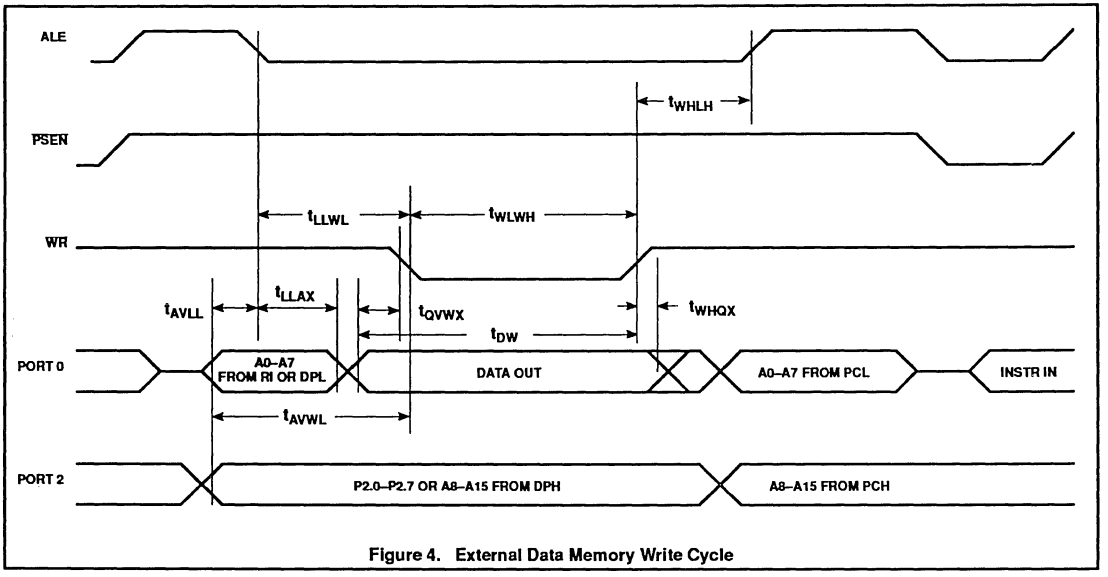


Figure 4. External Data Memory Write Cycle

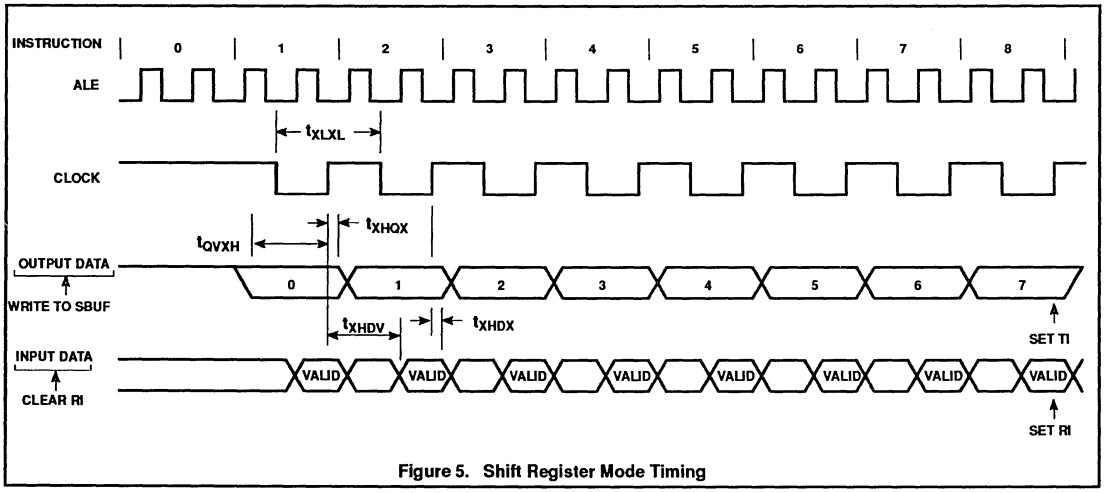


Figure 5. Shift Register Mode Timing

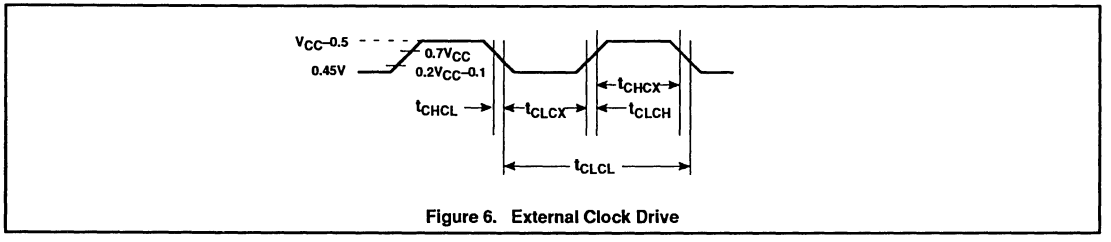


Figure 6. External Clock Drive

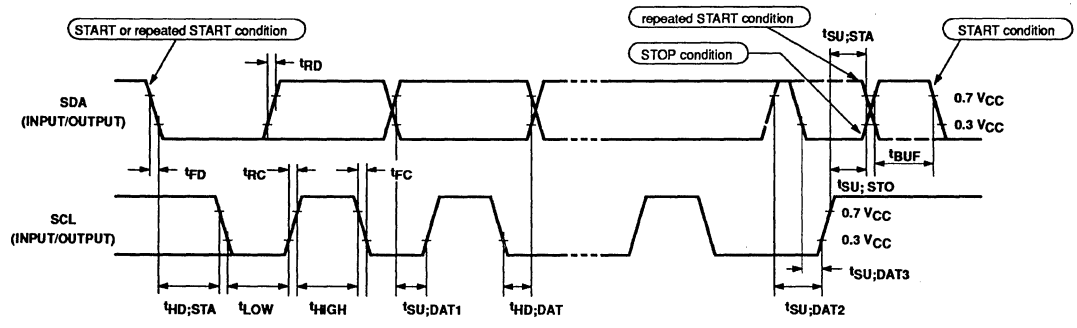
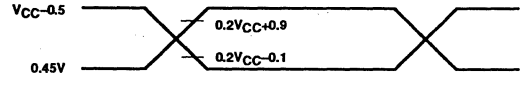
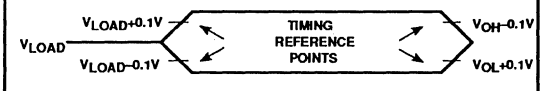


Figure 7. Timing SIO1 (I^2C) Interface



NOTE:
AC INPUTS DURING TESTING ARE DRIVEN AT V_{CC}–0.5 FOR A LOGIC '1' AND 0.45V FOR A LOGIC '0'. TIMING MEASUREMENTS ARE MADE AT V_{IH} MIN FOR A LOGIC '1' AND V_{IL} MAX FOR A LOGIC '0'.

Figure 8. AC Testing Input/Output



NOTE:
FOR TIMING PURPOSES, A PORT IS NO LONGER FLOATING WHEN A 100MV CHANGE FROM LOAD VOLTAGE OCCURS, AND BEGINS TO FLOAT WHEN A 100mV CHANGE FROM THE LOADED V_{OH}/V_{OL} LEVEL OCCURS.
$I_{OH}/I_{OL} \geq \pm 20mA$.

Figure 9. Float Waveform

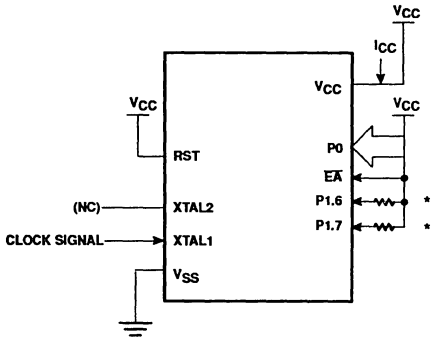


Figure 10. I_{CC} Test Condition, Active Mode
All other pins are disconnected

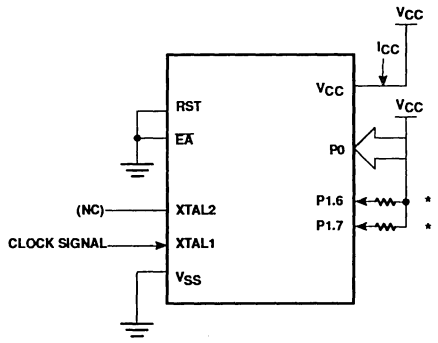


Figure 11. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

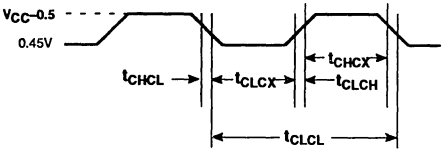


Figure 12. Clock Signal Waveform for I_{CC}
Tests in Active and Idle Modes
$t_{CLCL} = t_{CHCL} = 10ns$

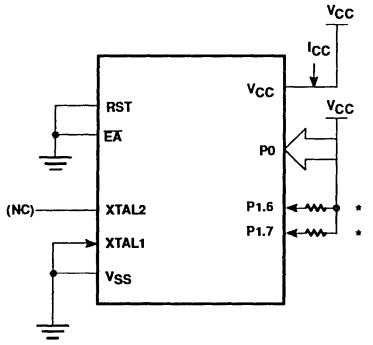


Figure 13. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected
V_{CC} = 2V to 5.5V

NOTE:

* Ports 1.6 and 1.7 should be connected to V_{CC} through resistors of sufficiently high value such that the sink current into these pins does not exceed the I_{OL1} specification.

EPROM CHARACTERISTICS

The 87C652 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C652 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C652 manufactured by Philips Components.

Table 3 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 14 and 15. Figure 16 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 14. Note that the 87C652 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 14. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 2 and 3 specified in Table 3 are held at the 'Program Code Data' levels indicated in Table 3. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 15.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 1FH, using the "Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25 pulse programming sequence using the 'Pgm Lock Bit' levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 1 and 2 as shown in Figure 16. The other pins are held at the 'Verify Code Data' levels indicated in Table 3. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 5H indicates manufactured by Philips

(031H) = 99H indicates 87C652

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 3, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient. Erasure leaves the array in an all 1s state.

Table 3. EPROM Programming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:

1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.

* ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

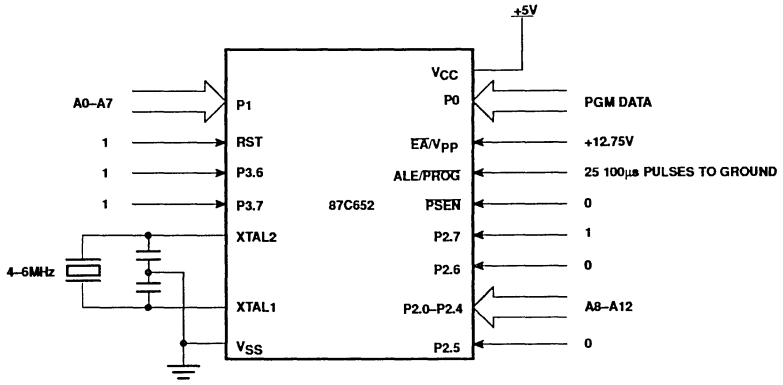


Figure 14. Programming Configuration

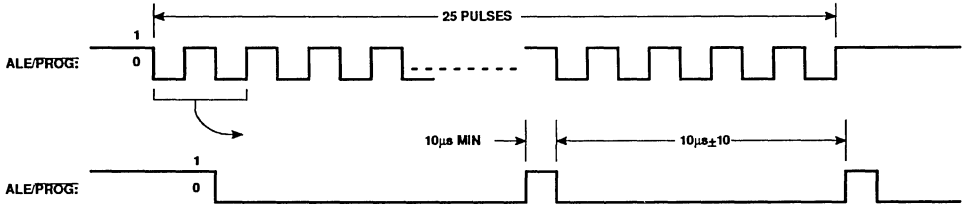


Figure 15. PROG Waveform

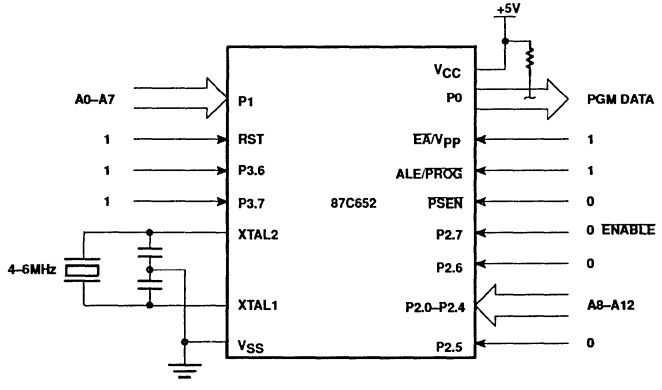


Figure 16. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_{amb} = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (see Figure 17)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

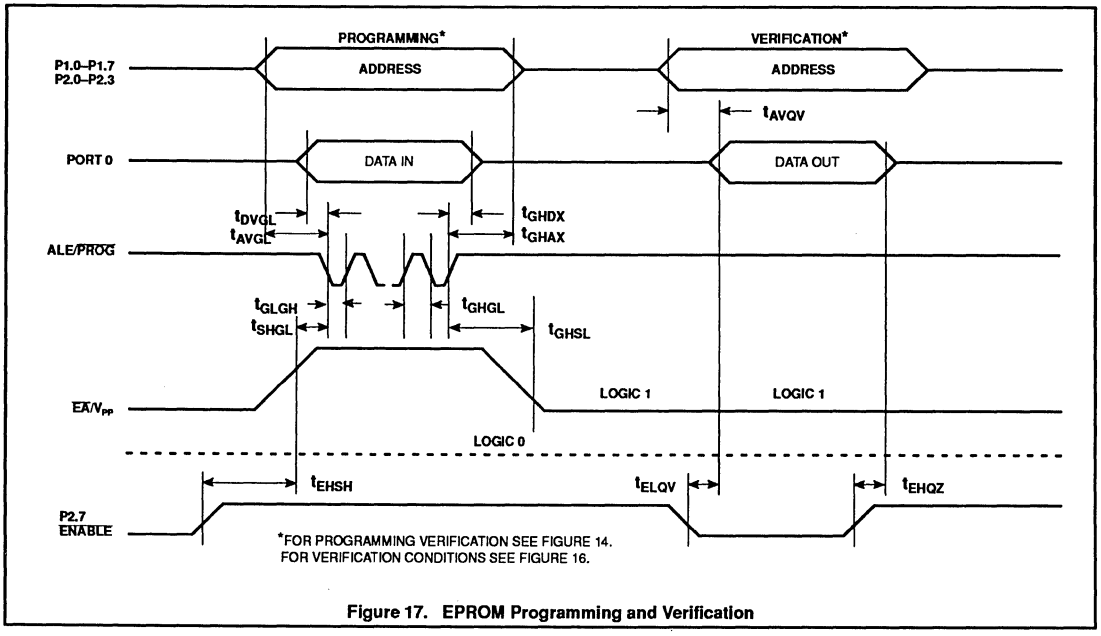


Figure 17. EPROM Programming and Verification

Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specifications defined by Philips.

CMOS single-chip 8-bit microcontroller — 83C654/87C654

DESCRIPTION

The 83C654/87C654 Single-Chip 8-Bit Microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 83C654/87C654 has the same instruction set as the 80C51. Two versions of the derivative exist:

83C654 — 16k bytes mask programmable ROM

87C654 — EPROM version

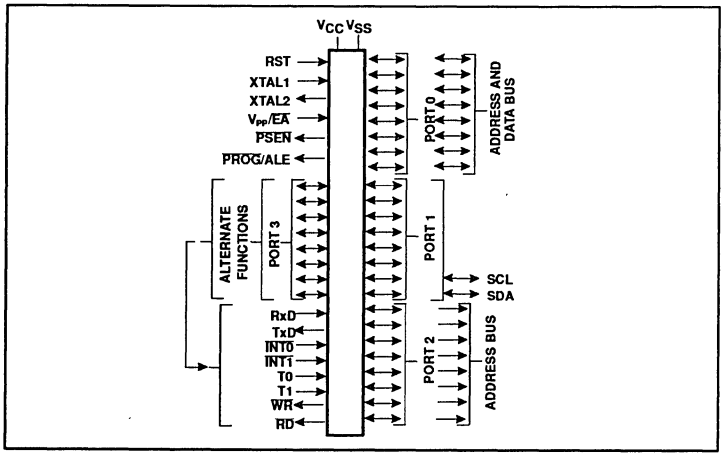
This device provides architectural enhancements that make it applicable in a variety of applications for general control systems. The 8XC654 contains a non-volatile 16k × 8 read-only program memory (83C654) EPROM (87C654), a volatile 256 × 8 read/write data memory, four 8-bit I/O ports, two 16-bit timer/event counters (identical to the timers of the 80C51), a multi-source, two-priority-level, nested interrupt structure, an I^2C interface, UART and on-chip oscillator and timing circuits. For systems that require extra capability, the 8XC654 can be expanded using standard TTL compatible memories and logic.

The device also functions as an arithmetic processor having facilities for both binary and BCD arithmetic plus bit-handling capabilities. The instruction set consists of over 100 instructions: 49 one-byte, 45 two-byte and 17 three-byte. With a 16(24)MHz crystal, 58% of the instructions are executed in 0.75(0.5)µs and 40% in 1.5(1)µs. Multiply and divide instructions require 3(2)µs.

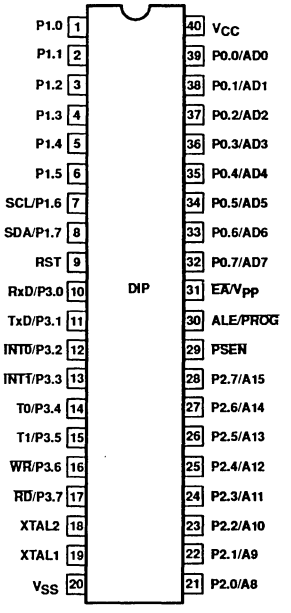
FEATURES

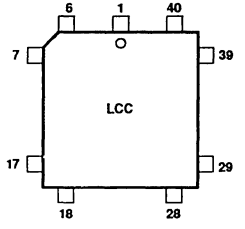
- 80C51 central processing unit
- 16k × 8 ROM expandable externally to 64k bytes
- 256 × 8 RAM, expandable externally to 64k bytes
- Two standard 16-bit timer/counters
- Four 8-bit I/O ports
- I^2C-bus serial I/O port with byte oriented master and slave functions
- Full-duplex UART facilities
- Power control modes
  - Idle mode
  - Power-down mode
- ROM code protection
- Five package styles
- Extended temperature ranges
- OTP package available
- Three speed ranges
  - 16MHz
  - 20MHz (87C654 only)
  - 24MHz (83C654 only)

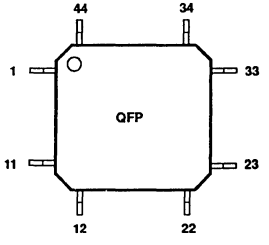
LOGIC SYMBOL



PIN CONFIGURATIONS







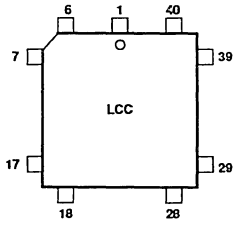
SEE PAGE 543 FOR QFP AND LCC PIN FUNCTIONS.

PART NUMBER SELECTION

PHILIPS PART ORDER NUMBER PART MARKING		SIGNETICS PART ORDER NUMBER			TEMPERATURE °C AND PACKAGE	FREQ.
ROMless[1]	ROM	ROMless[1]	ROM	EPROM		
P80C652FBP	P83C654FBP/xxx	S80C652-4N40	S83C654-4N40	S87C654-4N40	0 to +70, Plastic DIP	16MHz
				S87C654-4F40	0 to +70, Ceramic DIP with window	16MHz
P80C652FBA	P83C654FBA/xxx	S80C652-4A44	S83C654-4A44	S87C654-4A44	0 to +70, Plastic PLCC	16MHz
				S87C654-4K44	0 to +70, Ceramic CLCC with window	16MHz
P80C652FBB	P83C654FBB/xxx	S80C652-4B44	S83C654-4B44	S87C654–4B44	0 to +70, Plastic QFP	16MHz
P80C652FFP	P83C654FFP/xxx	S80C652–5N40	S83C654-5N40	S87C654-5N40	–40 to +85, Plastic DIP	16MHz
				S87C654-5F40	–40 to +85, Ceramic DIP with window	16MHz
P80C652FFA	P83C654FFA/xxx	S80C652–5A44	S83C654-5A44	S87C654-5A44	–40 to +85, Plastic PLCC	16MHz
				S87C654–5K44	–40 to +85, Ceramic CLCC with window	16MHz
P80C652FFB	P83C654FFB/xxx	S80C652–5B44	S83C654-5B44	S87C654-5B44	–40 to +85, Plastic QFP	16MHz
P80C652FHP	P83C654FHP/xxx	S80C652–6N40	S83C654-6N40		–40 to +125, Plastic DIP	16MHz
P80C652FHA	P83C654FHA/xxx	S80C652–6A44	S83C654-6A44		–40 to +125, Plastic PLCC	16MHz
P80C652FHB	P83C654FHB/xxx	S80C652–6B44	S83C654-6B44		–40 to +125, Plastic QFP	16MHz
				S87C654–7N40	0 to +70, Plastic DIP	20MHz
				S87C654–7F40	0 to +70, Ceramic DIP with window	20MHz
				S87C654–7A44	0 to +70, Plastic PLCC	20MHz
				S87C654–7K44	0 to +70, Ceramic CLCC with window	20MHz
				S87C654–7B44	0 to +70, Plastic QFP	20MHz
				S87C654–8N40	–40 to +85, Plastic DIP	20MHz
				S87C654–8F40	–40 to +85, Ceramic DIP with window	20MHz
				S87C654–8A44	–40 to +85, Plastic PLCC	20MHz
				S87C654–8K44	–40 to +85, Ceramic CLCC with window	20MHz
				S87C654–8B44	–40 to +85, Plastic QFP	20MHz
P80C652IBP	P83C654IBP/xxx	S80C652-AN40	S83C654-AN40		0 to +70, Plastic DIP	24MHz
P80C652IBA	P83C654IBA/xxx	S80C652-AA44	S83C654-AA44		0 to +70, Plastic PLCC	24MHz
P80C652IBB	P83C654IBB/xxx	S80C652-AB44	S83C654-AB44		0 to +70, Plastic QFP	24MHz
P80C652IFP	P83C654IFP/xxx	S80C652-BN40	S83C654-BN40		–40 to +85, Plastic DIP	24MHz
P80C652IFA	P83C654IFA/xxx	S80C652-BA44	S83C654-BA44		–40 to +85, Plastic PLCC	24MHz
P80C652IFB	P83C654IFB/xxx	S80C652-BB44	S83C654-BB44		–40 to +85, Plastic QFP	24MHz

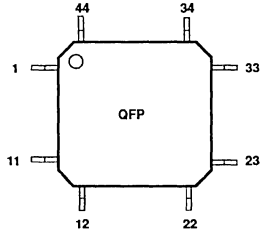
NOTES:
1. For full specification, see the 80C652/83C652/87C652 data sheet.
2. 83C654 frequency range is 1.2MHz – 16MHz or 1.2MHz – 24MHz.
3. 87C654 frequency range is 3.5MHz – 16MHz or 3.5MHz – 20MHz.
4. xxx denotes the ROM code number.

LCC PIN FUNCTIONS



Pin	Function	Pin	Function
1	NC	23	NC
2	P1.0	24	P2.0/A8
3	P1.1	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6/SCL	30	P2.6/A14
9	P1.7/SDA	31	P2.7/A15
10	RST	32	$\overline{PSEN}$
11	P3.0/RxD	33	ALE/$\overline{PROG}$
12	NC	34	NC
13	P3.1/TxD	35	$\overline{EA}$/V_{PP}
14	P3.2/$\overline{INT0}$	36	P0.7/AD7
15	P3.3/$\overline{INT1}$	37	P0.6/AD6
16	P3.4/T0	38	P0.5/AD5
17	P3.5/T1	39	P0.4/AD4
18	P3.6/$\overline{WR}$	40	P0.3/AD3
19	P3.7/$\overline{RD}$	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	V_{SS}	44	V_{CC}

QFP PIN FUNCTIONS

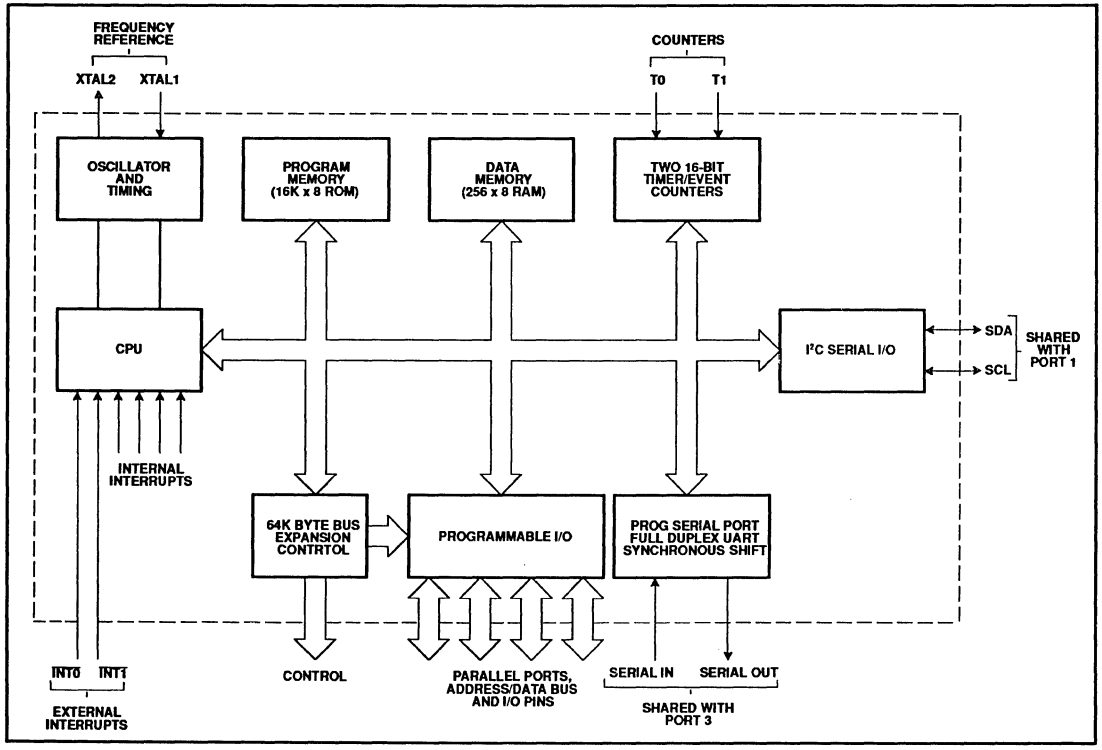


Pin	Function	Pin	Function
1	P1.5	23	P2.5/A13
2	P1.6/SCL	24	P2.6/A14
3	P1.7/SDA	25	P2.7/A15
4	RST	26	$\overline{PSEN}$
5	P3.0/RxD	27	ALE/$\overline{PROG}$
6	NC	28	NC
7	P3.1/TxD	29	$\overline{EA}$/V_{PP}
8	P3.2/$\overline{INT0}$	30	P0.7/AD7
9	P3.3/$\overline{INT1}$	31	P0.6/AD6
10	P3.4/T0	32	P0.5/AD5
11	P3.5/T1	33	P0.4/AD4
12	P3.6/$\overline{WR}$	34	P0.3/AD3
13	P3.7/$\overline{RD}$	35	P0.2/AD2
14	XTAL2	36	P0.1/AD1
15	XTAL1	37	P0.0/AD0
16	V_{SS}	38	V_{CC}
17	NC	39	NC
18	P2.0/A8	40	P1.0
19	P2.1/A9	41	P1.1
20	P2.2/A10	42	P1.2
21	P2.3/A11	43	P1.3
22	P2.4/A12	44	P1.4

NOTES TO QFP ONLY:

1. Due to EMC improvements, it is advised to connect pins 6, 28, 39 to V_{SS} on the 80C652/83C654.

BLOCK DIAGRAM



PIN DESCRIPTIONS

MNEMONIC	PIN NUMBER DIP	LCC	QFP	TYPE	NAME AND FUNCTION
V_{SS}	20	22	16	I	**Ground:** 0V reference.
V_{CC}	40	44	38	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–36	37–30	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 also outputs the code bytes during program verification in the 87C654. External pull-ups are required during program verification.
P1.0–P1.7	1–8	2–9	40–44, 1–3	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups, except P1.6 and P1.7 which are open drain. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 1 also receives the low-order address byte during program memory verification. Alternate functions include:
P1.6	7	8	2	I/O	**SCL:** I^2C-bus serial port clock line.
P1.7	8	9	3	I/O	**SDA:** I^2C-bus serial port data line.
P2.0–P2.7	21–28	24–31	18–25	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	5, 7–13	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 80C51 family, as listed below:
	10	11	5	I	**RxD (P3.0):** Serial input port
	11	13	7	O	**TxD (P3.1):** Serial output port
	12	14	8	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	9	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	10	I	**T0 (P3.4):** Timer 0 external input
	15	17	11	I	**T1 (P3.5):** Timer 1 external input
	16	18	12	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	13	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	4	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE/$\overline{PROG}$	30	33	27	I/O	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming.
$\overline{PSEN}$	29	32	26	O	**Program Store Enable:** The read strobe to external program memory. When the 87C654 is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}/V_{PP}$	31	35	29	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H and 3FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 3FFFH. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming.
XTAL1	19	21	15	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	14	O	**Crystal 2:** Output from the inverting oscillator amplifier.

NOTE:
To avoid "latch-up" effect at power-on, the voltage on any pin at any time must not be higher than V_{CC} + 0.5V or V_{SS} – 0.5V, respectively.

ROM CODE PROTECTION (83C654)

The 83C654 has an additional security feature. ROM code protection may be selected by setting a mask–programmable security bit (i.e., user dependent). This feature may be requested during ROM code submission. When selected, the ROM code is protected and cannot be read out at any time by any test mode or by any instruction in the external program memory space.

The MOVC instructions are the only instructions that have access to program code in the internal or external program memory. The $\overline{EA}$ input is latched during RESET and is "don't care" after RESET. This implementation prevents reading internal program code by switching from external program memory to internal program memory during a MOVC instruction or any other instruction that uses immediate data.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the Logic Symbol, page 541.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

Reset

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-on reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

Idle Mode

In the idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

Power-Down Mode

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. The control bits for the reduced power modes are in the special function register PCON. Table 1 shows the state of the I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Mode

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Serial Control Register (S1CON) – See Table 2

S1CON (D8H)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

Bits CR0, CR1 and CR2 determine the serial clock frequency that is generated in the master mode of operation.

Table 2. Serial Clock Rates

CR2	CR1	CR0	BIT FREQUENCY (kHz) AT f_{OSC} 6MHz	12MHz	16MHz	24MHz	f_{OSC} DIVIDED BY
0	0	0	23	47	62.5	94	256
0	0	1	27	54	71	107[1]	224
0	1	0	31.25	62.5	83.3	125[1]	192
0	1	1	37	75	100	150[1]	160
1	0	0	6.25	12.5	17	25	960
1	0	1	50	100	133[1]	200[1]	120
1	1	0	100	200[1]	267[1]	400[1]	60
1	1	1	0.25 < 62.5	0.5 < 62.5	0.67 < 56	0.98 < 50	96 × (256 – (reload value Timer 1)) (Reload value range: 0 – 254 in mode 2)

NOTES:

1. These frequencies exceed the upper limit of 100kHz of the I²C-bus specification and cannot be used in an I²C-bus application.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ to V_{SS} (87C654 only)	–0.5 to + 13	V
Voltage on any other pin to V_{SS}	–0.5 to + 6.5	V
Input, output current on any single pin	±5	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DEVICE SPECIFICATIONS

	SUPPLY VOLTAGE (V)		FREQUENCY (MHz)		TEMPERATURE RANGE
TYPE	MIN.	MAX.	MIN.	MAX.	(°C)
P83C654FB	4.0	6.0	1.2	16	0 to +70
S87C654–4	4.5	5.5	3.5	16	0 to +70
P83C654FF	4.0	6.0	1.2	16	–40 to +85
S87C654–5	4.5	5.5	3.5	16	–40 to +85
P83C654FH	4.5	5.5	1.2	16	–40 to +125
S87C654–7	4.5	5.5	3.5	20	0 to +70
S87C654–8	4.5	5.5	3.5	20	–40 to +85
P83C654IB	4.5	5.5	1.2	24	0 to +70
P83C654IF	4.5	5.5	1.2	24	–40 to +85

DC ELECTRICAL CHARACTERISTICS

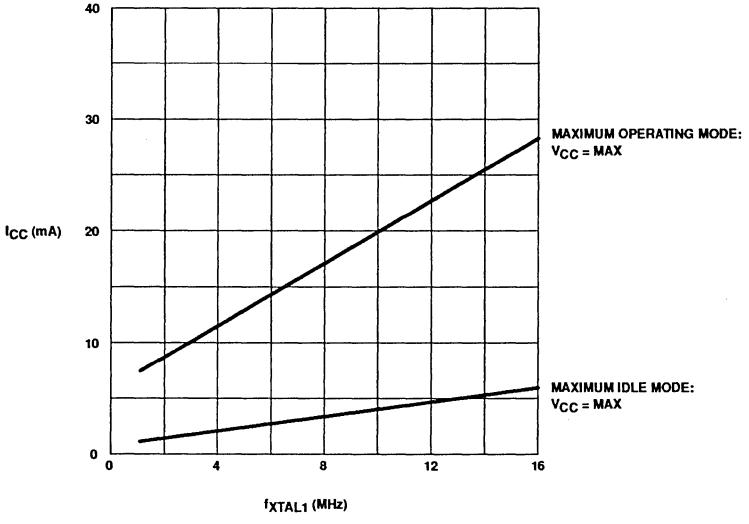
V_{SS} = 0V

SYMBOL	PARAMETER	PART TYPE	TEST CONDITIONS	LIMITS MIN.	LIMITS MAX.	UNIT
V_{IL}	Input low voltage, except $\overline{EA}$, P1.6/SCL, P1.7/SDA	0 to +70°C –40 to +85°C –40 to +125°C		–0.5 –0.5 –0.5	$0.2V_{CC}$–0.1 $0.2V_{CC}$–0.15 $0.2V_{CC}$–0.25	V V V
V_{IL1}	Input low voltage to $\overline{EA}$	0 to +70°C –40 to +85°C –40 to +125°C		–0.5 –0.5 –0.5	$0.2V_{CC}$–0.3 $0.2V_{CC}$–0.35 $0.2V_{CC}$–0.45	V V V
V_{IL2}	Input low voltage to P1.6/SCL, P1.7/SDA[6]			–0.5	$0.3V_{CC}$	V
V_{IH}	Input high voltage, except XTAL1, RST, P1.6/SCL, P1.7/SDA	0 to +70°C –40 to +85°C –40 to +125°C		$0.2V_{CC}$+0.9 $0.2V_{CC}$+1.0 $0.2V_{CC}$+1.0	V_{CC}+0.5 V_{CC}+0.5 V_{CC}+0.5	V V V
V_{IH1}	Input high voltage, XTAL1, RST	0 to +70°C –40 to +85°C –40 to +125°C		$0.7V_{CC}$ $0.7V_{CC}$+0.1 $0.7V_{CC}$+0.1	V_{CC}+0.5 V_{CC}+0.5 V_{CC}+0.5	V V V
V_{IH2}	Input high voltage, P1.6/SCL, P1.7/SDA[6]			$0.7V_{CC}$	6.0	V
V_{OL}	Output low voltage, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA		I_{OL} = 1.6mA[8, 9]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$		I_{OL} = 3.2mA[8, 9]		0.45	V
V_{OL2}	Output low voltage, P1.6/SCL, P1.7/SDA		I_{OL} = 3.0mA		0.4	V
V_{OH}	Output high voltage, ports 1, 2, 3		I_{OH} = –60µA I_{OH} = –25µA I_{OH} = –10µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$		V V V
V_{OH1}	Output high voltage; port 0 in external bus mode, ALE, $\overline{PSEN}$, RST[10]		I_{OH} = –400µA I_{OH} = –150µA I_{OH} = –40µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$		V V V
I_{IL}	Logical 0 input current, ports 1, 2, 3, 4, except P1.6/SCL, P1.7/SDA	0 to +70°C –40 to +85°C –40 to +125°C	V_{IN} = 0.45V		–50 –75 –75	µA µA µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA	0 to +70°C –40 to +85°C –40 to +125°C	See note 7		–650 –750 –750	µA µA µA
I_{L1}	Input leakage current, port 0		0.45V < V_I < V_{CC}		±10	µA
I_{L2}	Input leakage current, P1.6/SCL, P1.7/SDA		0V < V_I < 6.0V 0V < V_{CC} < 6.0V		±10	µA µA
I_{CC}	Power supply current:		See note 1			
	Active mode @ 16MHz (83C654)[2, 11]		V_{CC}=6.0V		28	mA
	Active mode @ 16MHz (87C654)[2]				25	mA
	Active mode @ 24MHz (80/83C654)[2, 11]		V_{CC}=5.5V		35	mA
	Idle mode @ 16MHz (83C654)[3, 11]				6	mA
	Idle mode @ 16MHz (87C654)[3]				6	mA
	Idle mode @ 24MHz (80/83C654)[3, 11]				7	mA
	Power down mode[4, 5]				50	µA
	Power down mode (87C654)[4, 5]	–40 to +85°C			135	µA
	Power down mode (80/83C654)[4, 5]	–40 to +125°C			100	µA
R_{RST}	Internal reset pull-down resistor			50	150	kΩ
C_{IO}	Pin capacitance		Freq.=1MHz		10	pF

NOTES: See Next Page.

NOTES FOR DC ELECTRICAL CHARACTERISTICS:

1. See Figures 9 through 12 for I_{CC} test conditions.
2. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{CC} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = P1.6 = P1.7 = V_{CC}; f_{CLK} = 16MHz. See Figure 9.
3. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 10ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{CC} - 0.5V$; XTAL2 not connected; Port 0 = P1.6 = P1.7 = V_{CC}; $\overline{EA}$ = RST = V_{SS}; f_{CLK} = 16MHz. See Figure 10.
4. The power-down current is measured with all output pins disconnected; XTAL2 not connected; Port 0 = P1.6 = P1.7 = V_{CC}; $\overline{EA}$ = RST = V_{SS}. See Figure 12.
5. $2V \leq V_{PD} \leq V_{CC}$max.
6. The input threshold voltage of P1.6 and P1.7 (SIO1) meets the I^2C specification, so an input voltage below $0.3V_{CC}$ will be recognized as a logic 0 while an input voltage above $0.7V_{CC}$ will be recognized as a logic 1.
7. Pins of ports 1 , 2, and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
8. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. I_{OL} can exceed these conditions provided that no single output sinks more than 5mA and no more than two outputs exceed the test conditions.
9. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows: Maximum I_{OL} = 10mA per port pin; Maximum I_{OL} = 26mA total for Port 0; Maximum I_{OL} = 15mA total for Ports 1, 2, and 3; Maximum I_{OL} = 71mA total for all output pins. If I_{OL} exceeds the test conditions, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
10. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
11. I_{CCMAX} for the 80/83C654 at the other frequencies can be derived from Figure 1, where FREQ is the external oscillator frequency in MHz. I_{CCMAX} is given in mA.



VALID ONLY WITHIN FREQUENCY SPECIFICATIONS OF DEVICE UNDER TEST.

Figure 1. I_{CC} vs. Frequency

AC ELECTRICAL CHARACTERISTICS[1, 2]

SYMBOL	FIGURE	PARAMETER	16MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	2	Oscillator frequency			1.2 (3.5)	16	MHz
t_{LHLL}	2	ALE pulse width	85		$2t_{CLCL}-40$		ns
t_{AVLL}	2	Address valid to ALE low	8		$t_{CLCL}-55$		ns
t_{LLAX}	2	Address hold after ALE low	28		$t_{CLCL}-35$		ns
t_{LLIV}	2	ALE low to valid instruction in		150		$4t_{CLCL}-100$	ns
t_{LLPL}	2	ALE low to $\overline{PSEN}$ low	23		$t_{CLCL}-40$		ns
t_{PLPH}	2	$\overline{PSEN}$ pulse width	143		$3t_{CLCL}-45$		ns
t_{PLIV}	2	$\overline{PSEN}$ low to valid instruction in		83		$3t_{CLCL}-105$	ns
t_{PXIX}	2	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	2	Input instruction float after $\overline{PSEN}$		38		$t_{CLCL}-25$	ns
t_{AVIV}	2	Address to valid instruction in		208		$5t_{CLCL}-105$	ns
t_{PLAZ}	2	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{AVLL}	3, 4	Address valid to ALE low	28		$t_{CLCL}-35$		ns
t_{RLRH}	3, 4	$\overline{RD}$ pulse width	275		$6t_{CLCL}-100$		ns
t_{WLWH}	3, 4	$\overline{WR}$ pulse width	275		$6t_{CLCL}-100$		ns
t_{RLDV}	3, 4	$\overline{RD}$ low to valid data in		148		$5t_{CLCL}-165$	ns
t_{RHDX}	3, 4	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	3, 4	Data float after $\overline{RD}$		55		$2t_{CLCL}-70$	ns
t_{LLDV}	3, 4	ALE low to valid data in		350		$8t_{CLCL}-150$	ns
t_{AVDV}	3, 4	Address to valid data in		398		$9t_{CLCL}-165$	ns
t_{LLWL}	3, 4	ALE low to $\overline{RD}$ or $\overline{WR}$ low	138	238	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	3, 4	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	120		$4t_{CLCL}-130$		ns
t_{QVWX}	3, 4	Data valid to $\overline{WR}$ transition	3		$t_{CLCL}-60$		ns
t_{DW}	3, 4	Data setup time before $\overline{WR}$	288		$7t_{CLCL}-150$		ns
t_{WHQX}	3, 4	Data hold after $\overline{WR}$	13		$t_{CLCL}-50$		ns
t_{RLAZ}	3, 4	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	3, 4	$\overline{RD}$ or $\overline{WR}$ high to ALE high	23	103	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
Shift Register							
t_{XLXL}	5	Serial port clock cycle time[3]	0.75		$12t_{CLCL}$		µs
t_{QVXH}	5	Output data setup to clock rising edge[3]	492		$10t_{CLCL}-133$		ns
t_{XHQX}	5	Output data hold after clock rising edge[3]	80		$2t_{CLCL}-117$		ns
t_{XHDX}	5	Input data hold after clock rising edge[3]	0		0		ns
t_{XHDV}	5	Clock rising edge to input data valid[3]		492		$10t_{CLCL}-133$	ns
External Clock							
t_{CHCX}	6	High time[3]	20		20	$t_{CLCL}-t_{LOW}$	ns
t_{CLCX}	6	Low time[3]	20		20	$t_{CLCL}-t_{HIGH}$	ns
t_{CLCH}	6	Rise time[3]		20		20	ns
t_{CHCL}	6	Fall time[3]		20		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. These values are characterized but not 100% production tested.

AC ELECTRICAL CHARACTERISTICS[1, 2]

SYMBOL	FIGURE	PARAMETER	24MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	2	Oscillator frequency			1.2 (3.5)	24 (20)	MHz
t_{LHLL}	2	ALE pulse width	43		$2t_{CLCL}-40$		ns
t_{AVLL}	2	Address valid to ALE low	17		$t_{CLCL}-25$		ns
t_{LLAX}	2	Address hold after ALE low	17		$t_{CLCL}-25$		ns
t_{LLIV}	2	ALE low to valid instruction in		102		$4t_{CLCL}-65$	ns
t_{LLPL}	2	ALE low to $\overline{PSEN}$ low	17		$t_{CLCL}-25$		ns
t_{PLPH}	2	$\overline{PSEN}$ pulse width	80		$3t_{CLCL}-45$		ns
t_{PLIV}	2	$\overline{PSEN}$ low to valid instruction in		65		$3t_{CLCL}-60$	ns
t_{PXIX}	2	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	2	Input instruction float after $\overline{PSEN}$		17		$t_{CLCL}-25$	ns
t_{AVIV}	2	Address to valid instruction in		128		$5t_{CLCL}-80$	ns
t_{PLAZ}	2	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{AVLL}	3, 4	Address valid to ALE low	17		$t_{CLCL}-25$		ns
t_{RLRH}	3, 4	$\overline{RD}$ pulse width	150		$6t_{CLCL}-100$		ns
t_{WLWH}	3, 4	$\overline{WR}$ pulse width	150		$6t_{CLCL}-100$		ns
t_{RLDV}	3, 4	$\overline{RD}$ low to valid data in		118		$5t_{CLCL}-90$	ns
t_{RHDX}	3, 4	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	3, 4	Data float after $\overline{RD}$		55		$2t_{CLCL}-28$	ns
t_{LLDV}	3, 4	ALE low to valid data in		183		$8t_{CLCL}-150$	ns
t_{AVDV}	3, 4	Address to valid data in		210		$9t_{CLCL}-165$	ns
t_{LLWL}	3, 4	ALE low to $\overline{RD}$ or $\overline{WR}$ low	75	175	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	3, 4	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	92		$4t_{CLCL}-75$		ns
t_{QVWX}	3, 4	Data valid to $\overline{WR}$ transition	12		$t_{CLCL}-30$		ns
t_{DW}	3, 4	Data setup time before $\overline{WR}$	162		$7t_{CLCL}-130$		ns
t_{WHQX}	3, 4	Data hold after $\overline{WR}$	17		$t_{CLCL}-25$		ns
t_{RLAZ}	3, 4	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	3, 4	$\overline{RD}$ or $\overline{WR}$ high to ALE high	17	67	$t_{CLCL}-25$	$t_{CLCL}+25$	ns
Shift Register							
t_{XLXL}	5	Serial port clock cycle time[3]	0.5		$12t_{CLCL}$		µs
t_{QVXH}	5	Output data setup to clock rising edge[3]	283		$10t_{CLCL}-133$		ns
t_{XHQX}	5	Output data hold after clock rising edge[3]	23		$2t_{CLCL}-60$		ns
t_{XHDX}	5	Input data hold after clock rising edge[3]	0		0		ns
t_{XHDV}	5	Clock rising edge to input data valid[3]		283		$10t_{CLCL}-133$	ns
External Clock							
t_{CHCX}	6	High time[3]	17		17	$t_{CLCL}-t_{LOW}$	ns
t_{CLCX}	6	Low time[3]	17		17	$t_{CLCL}-t_{HIGH}$	ns
t_{CLCH}	6	Rise time[3]		20		20	ns
t_{CHCL}	6	Fall time[3]		20		20	ns

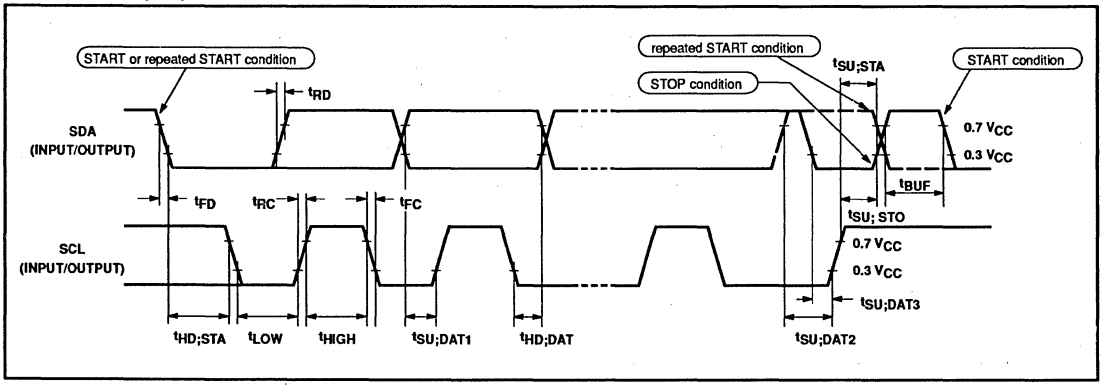
NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. These values are characterized but not 100% production tested.

AC ELECTRICAL CHARACTERISTICS – I²C INTERFACE

SYMBOL	PARAMETER	INPUT	OUTPUT
SCL TIMING CHARACTERISTICS			
$t_{HD;STA}$	START condition hold time	≥ 14 t_{CLCL}	> 4.0µs[1]
t_{LOW}	SCL LOW time	≥ 16 t_{CLCL}	> 4.7µs[1]
t_{HIGH}	SCL HIGH time	≥ 14 t_{CLCL}	> 4.0µs[1]
t_{RC}	SCL rise time	≤ 1µs	–[2]
t_{FC}	SCL fall time	≤ 0.3µs	< 0.3µs[3]
SDA TIMING CHARACTERISTICS			
$t_{SU;DAT1}$	Data set-up time	≥ 250ns	> 20 t_{CLCL} – t_{RD}
$t_{SU;DAT2}$	SDA set-up time (before rep. START cond.)	≥ 250ns	> 1µs[1]
$t_{SU;DAT3}$	SDA set-up time (before STOP cond.)	≥ 250ns	> 8 t_{CLCL}
$t_{HD;DAT}$	Data hold time	≥ 0ns	> 8 t_{CLCL} – t_{FC}
$t_{SU;STA}$	Repeated START set-up time	≥ 14 t_{CLCL}	> 4.7µs[1]
$t_{SU;STO}$	STOP condition set-up time	≥ 14 t_{CLCL}	> 4.0µs[1]
t_{BUF}	Bus free time	≥ 14 t_{CLCL}	> 4.7µs[1]
t_{RD}	SDA rise time	≤ 1µs	–[2]
t_{FD}	SDA fall time	≤ 0.3µs	< 0.3µs[3]

NOTES:
1. At 100 kbit/s. At other bit rates this value is inversely proportional to the bit-rate of 100 kbit/s.
2. Determined by the external bus-line capacitance and the external bus-line pull-resistor, this must be < 1µs.
3. Spikes on the SDA and SCL lines with a duration of less than 3 t_{CLCL} will be filtered out. Maximum capacitance on bus-lines SDA and SCL = 400pF.
4. $t_{CLCL} = 1/f_{OSC}$ = one oscillator clock period at pin XTAL1. For 62ns (42ns) < t_{CLCL} < 285ns (16MHz (24MHz) > f_{OSC} > 3.5MHz) the SI01 interface meets the I²C-bus specification for bit-rates up to 100 kbit/s.

TIMING SIO1 (I²C) INTERFACE



EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

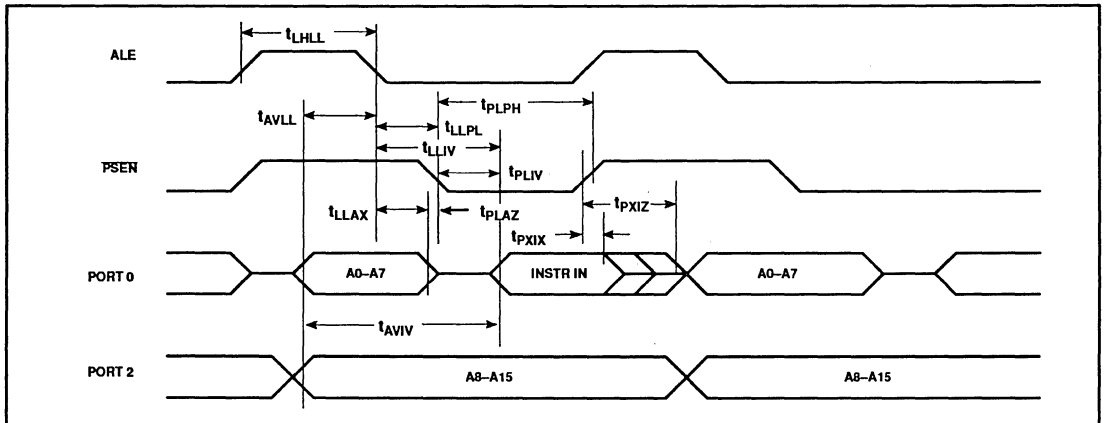


Figure 2. External Program Memory Read Cycle

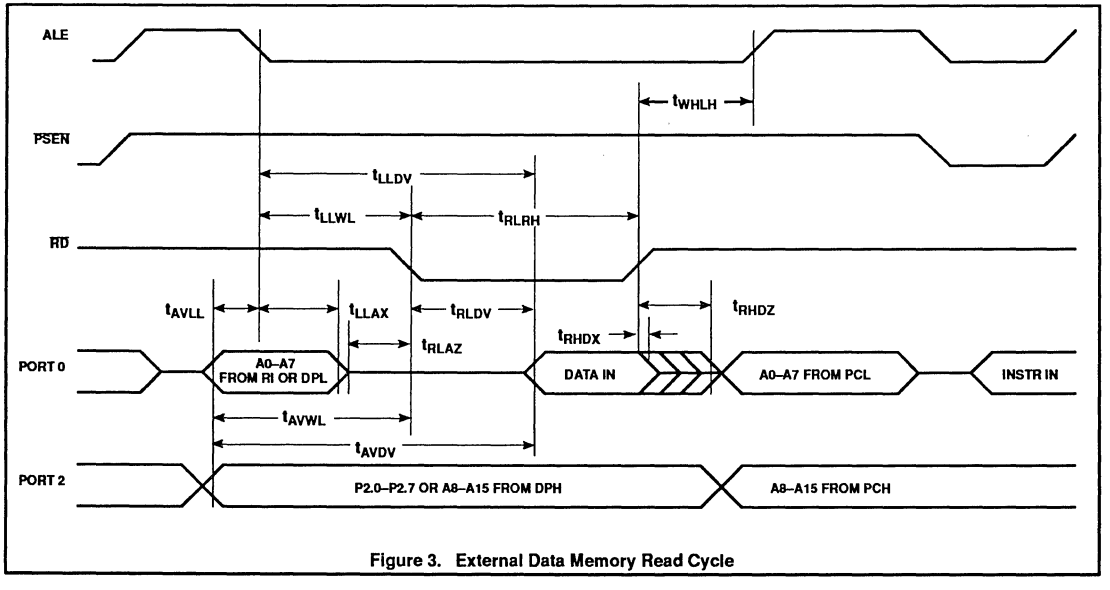


Figure 3. External Data Memory Read Cycle

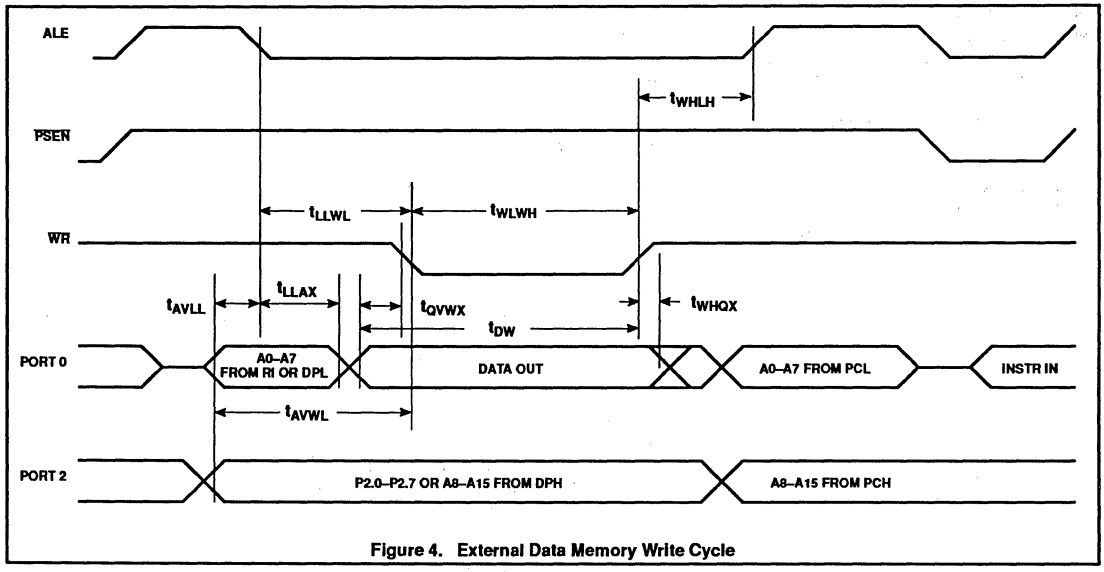


Figure 4. External Data Memory Write Cycle

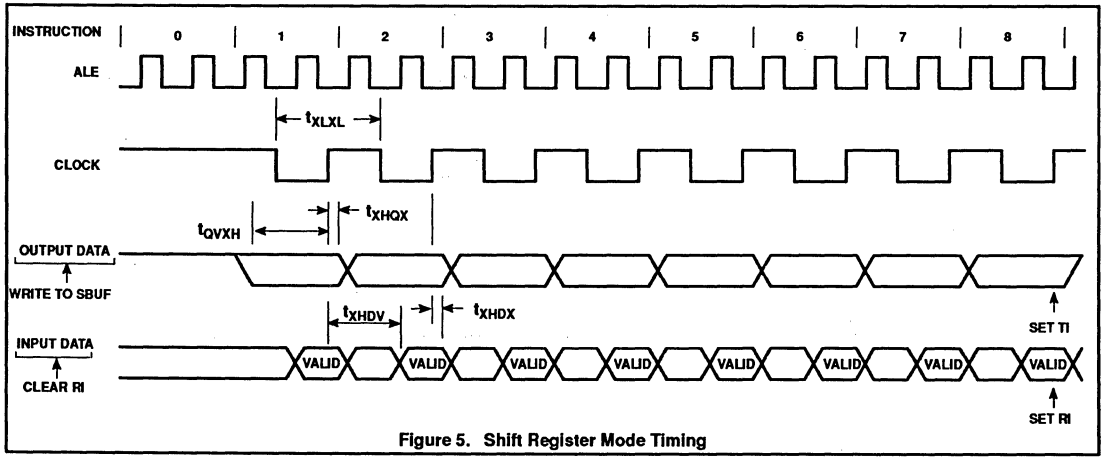


Figure 5. Shift Register Mode Timing

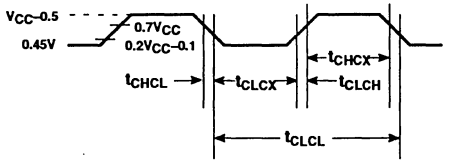
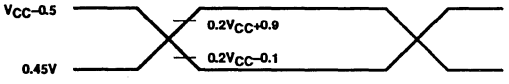
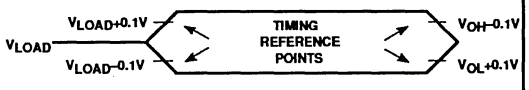


Figure 6. External Clock Drive



NOTE:
AC INPUTS DURING TESTING ARE DRIVEN AT V_{CC}–0.5 FOR A LOGIC '1' AND 0.45V FOR A LOGIC '0'. TIMING MEASUREMENTS ARE MADE AT V_{IH} MIN FOR A LOGIC '1' AND V_{IL} MAX FOR A LOGIC '0'.

Figure 7. AC Testing Input/Output



NOTE:
FOR TIMING PURPOSES, A PORT IS NO LONGER FLOATING WHEN A 100MV CHANGE FROM LOAD VOLTAGE OCCURS, AND BEGINS TO FLOAT WHEN A 100mV CHANGE FROM THE LOADED V_{OH}/V_{OL} LEVEL OCCURS. $I_{OH}/I_{OL} \geq \pm$ 20mA.

Figure 8. Float Waveform

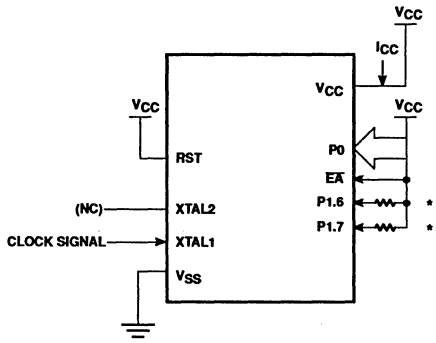


Figure 9. I_{CC} Test Condition, Active Mode
All other pins are disconnected

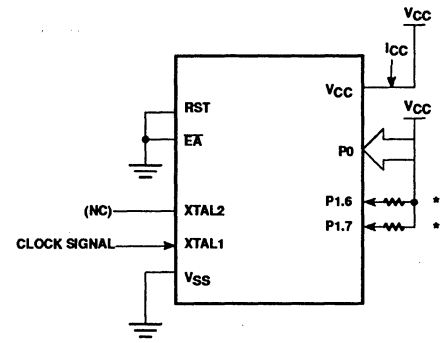


Figure 10. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

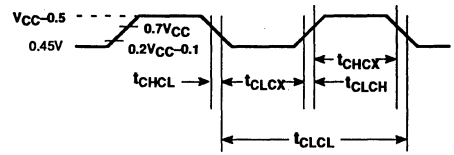


Figure 11. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 10ns$

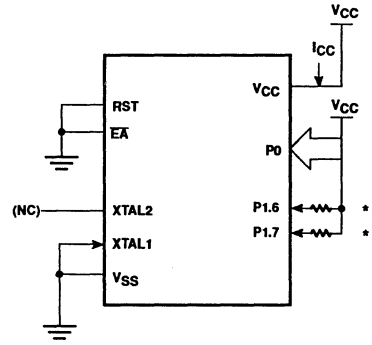


Figure 12. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

NOTE:
* Ports 1.6 and 1.7 should be connected to V_{CC} through resistors of sufficiently high value such that the sink current into these pins does not exceed the I_{OL1} specification.

EPROM CHARACTERISTICS

The 87C654 is programmed by using a modified Quick-Pulse Programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/$\overline{PROG}$ pulses.

The 87C654 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as an 87C654 manufactured by Philips Components.

Table 3 shows the logic levels for reading the signature byte, and for programming the program memory, the encryption table, and the lock bits. The circuit configuration and waveforms for quick-pulse programming are shown in Figures 13 and 14. Figure 15 shows the circuit configuration for normal program memory verification.

Quick-Pulse Programming

The setup for microcontroller quick-pulse programming is shown in Figure 13. Note that the 87C654 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers.

The address of the EPROM location to be programmed is applied to ports 1 and 2, as shown in Figure 13. The code byte to be programmed into that location is applied to port 0. RST, $\overline{PSEN}$ and pins of ports 2 and 3 specified in Table 3 are held at the 'Program Code Data' levels indicated in Table 3. The ALE/$\overline{PROG}$ is pulsed low 25 times as shown in Figure 14.

To program the encryption table, repeat the 25 pulse programming sequence for addresses 0 through 1FH, using the 'Pgm Encryption Table' levels. Do not forget that after the encryption table is programmed, verification cycles will produce only encrypted data.

To program the lock bits, repeat the 25 pulse programming sequence using the 'Pgm Lock Bit' levels. After one lock bit is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

Note that the $\overline{EA}$/V_{PP} pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be well regulated and free of glitches and overshoot.

Program Verification

If lock bit 2 has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory locations to be read is applied to ports 1 and 2 as shown in Figure 15. The other pins are held at the 'Verify Code Data' levels indicated in Table 3. The contents of the address location will be emitted on port 0. External pull-ups are required on port 0 for this operation.

If the encryption table has been programmed, the data presented at port 0 will be the exclusive NOR of the program byte with one of the encryption bytes. The user will have to know the encryption table contents in order to correctly decode the verification data. The encryption table itself cannot be read out.

Reading the Signature Bytes

The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 need to be pulled to a logic low. The values are:

(030H) = 15H indicates manufactured by Philips

(031H) = 99H indicates 87C654

Program/Verify Algorithms

Any algorithm in agreement with the conditions listed in Table 3, and which satisfies the timing specifications, is suitable.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Fluorglas part number 2345–5, or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient. Erasure leaves the array in an all 1s state.

Table 3. EPROM Programming Modes

MODE	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.7	P2.6	P3.7	P3.6
Read signature	1	0	1	1	0	0	0	0
Program code data	1	0	0*	V_{PP}	1	0	1	1
Verify code data	1	0	1	1	0	0	1	1
Pgm encryption table	1	0	0*	V_{PP}	1	0	1	0
Pgm lock bit 1	1	0	0*	V_{PP}	1	1	1	1
Pgm lock bit 2	1	0	0*	V_{PP}	1	1	0	0

NOTES:

1. '0' = Valid low for that pin, '1' = valid high for that pin.
2. V_{PP} = 12.75V ±0.25V.
3. V_{CC} = 5V±10% during programming and verification.

* ALE/$\overline{PROG}$ receives 25 programming pulses while V_{PP} is held at 12.75V. Each programming pulse is low for 100µs (±10µs) and high for a minimum of 10µs.

™Trademark phrase of Intel Corporation.

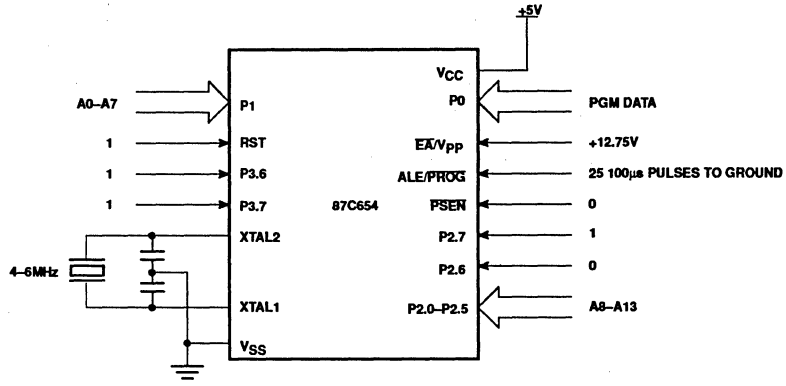


Figure 13. Programming Configuration

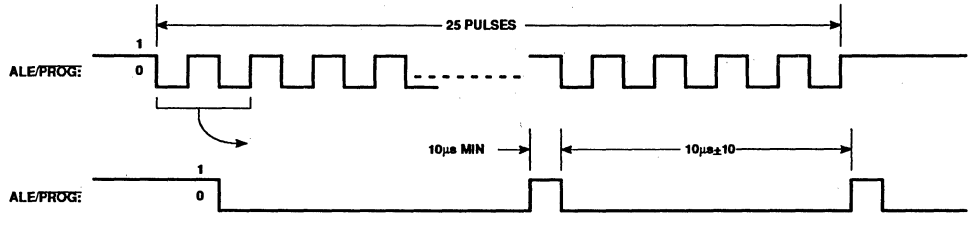


Figure 14. PROG Waveform

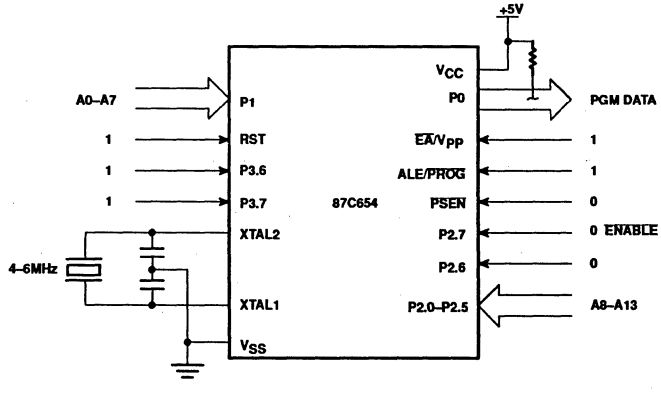


Figure 15. Program Verification

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_{amb} = 21°C to +27°C, V_{CC} = 5V±10%, V_{SS} = 0V (See Figure 16)

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{PP}	Programming supply voltage	12.5	13.0	V
I_{PP}	Programming supply current		50	mA
$1/t_{CLCL}$	Oscillator frequency	4	6	MHz
t_{AVGL}	Address setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHAX}	Address hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data setup to $\overline{PROG}$ low	$48t_{CLCL}$		
t_{GHDX}	Data hold after $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) high to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} setup to $\overline{PROG}$ low	10		µs
t_{GHSL}	V_{PP} hold after $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ width	90	110	µs
t_{AVQV}	Address to data valid		$48t_{CLCL}$	
t_{ELQZ}	$\overline{ENABLE}$ low to data valid		$48t_{CLCL}$	
t_{EHQZ}	Data float after $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHGL}	$\overline{PROG}$ high to $\overline{PROG}$ low	10		µs

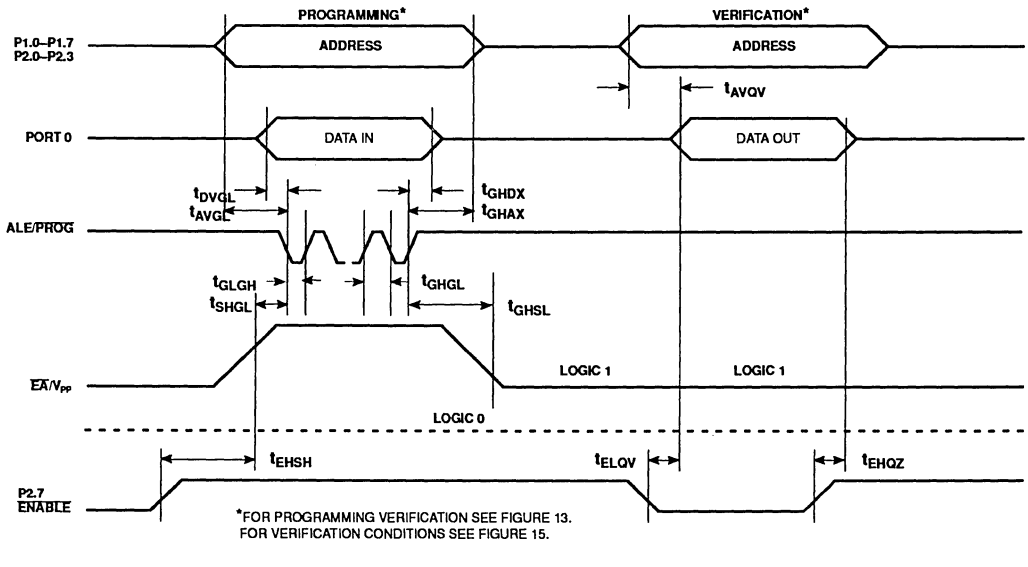


Figure 16. EPROM Programming and Verification

Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specifications defined by Philips.

CMOS single-chip 8-bit microcontroller with Electromagnetic Compatibility improvements

80CE654/83CE654

DESCRIPTION

The 83CE654 Single-Chip 8-Bit Microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 83CE654 has the same instruction set as the 80C51. Two versions of the derivative exist:

83CE654 — 16k bytes mask programmable ROM, 256 bytes RAM

80CE654 — ROMless version of the 83CE654

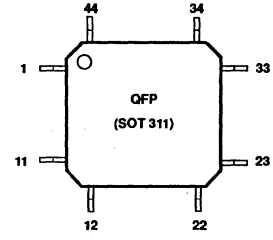
This device provides architectural enhancements that make it applicable in a variety of applications for general control systems. The 8XCE654 contains a non-volatile 16k × 8 read-only program memory (83CE654), a volatile 256 × 8 read/write data memory, four 8-bit I/O ports, two 16-bit timer/event counters (identical to the timers of the 80C51), a multi-source, two-priority-level, nested interrupt structure, an I²C interface, UART and on-chip oscillator and timing circuits. For systems that require extra capability, the 8XCE654 can be expanded using standard TTL compatible memories and logic.

The device also functions as an arithmetic processor having facilities for both binary and BCD arithmetic plus bit-handling capabilities. The instruction set consists of over 100 instructions: 49 one-byte, 45 two-byte and 17 three-byte. With a 16MHz crystal, 58% of the instructions are executed in 0.75µs and 40% in 1.5µs. Multiply and divide instructions require 3µs.

FEATURES

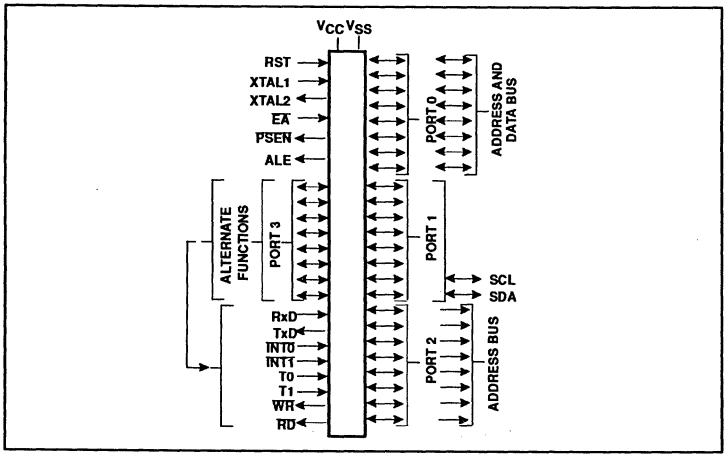
- 80C51 central processing unit
- 16k × 8 ROM expandable externally to 64k bytes
- 256 × 8 RAM, expandable externally to 64k bytes
- Two standard 16-bit timer/counters
- Four 8-bit I/O ports
- I²C-bus serial I/O port with byte oriented master and slave functions
- Full-duplex UART facilities
- ROM code protection
- XTAL frequencey range: 1.2MHz to 16MHz
- Software enable/disable of ALE output pulse
- Electromagnetic compatibility improvements
- Operating ambient temperature range:
  - P83CE654 FBB T_{amb} 0°C to +70°C
  - P83CE654 FFB T_{amb} –40°C to +85°C

PIN CONFIGURATION



Pin	Function	Pin	Function
1	P1.5	23	P2.5/A13
2	P1.6/SCL	24	P2.6/A14
3	P1.7/SDA	25	P2.7/A15
4	RST	26	$\overline{PSEN}$
5	P3.0/RxD	27	ALE
6	V_{SS4}	28	V_{SS2}
7	P3.1/TxD	29	$\overline{EA}$
8	P3.2/$\overline{INT0}$	30	P0.7/AD7
9	P3.3/$\overline{INT1}$	31	P0.6/AD6
10	P3.4/T0	32	P0.5/AD5
11	P3.5/T1	33	P0.4/AD4
12	P3.6/$\overline{WR}$	34	P0.3/AD3
13	P3.7/$\overline{RD}$	35	P0.2/AD2
14	XTAL2	36	P0.1/AD1
15	XTAL1	37	P0.0/AD0
16	V_{SS1}	38	V_{CC2}
17	V_{CC1}	39	V_{SS3}
18	P2.0/A8	40	P1.0
19	P2.1/A9	41	P1.1
20	P2.2/A10	42	P1.2
21	P2.3/A11	43	P1.3
22	P2.4/A12	44	P1.4

LOGIC SYMBOL



CMOS single-chip 8-bit microcontroller with Electromagnetic Compatibility improvements

80CE654/83CE654

PART NUMBER SELECTION

ROMless	ROM	TEMPERATURE °C AND PACKAGE	FREQUENCY
P80CE654FBB	P83CE654FBB	0 to +70, plastic QFP	1.2MHz to 16MHz
P80CE654FFB	P83CE654FFB	–40 to +85, plastic QFP	1.2MHz to 16MHz

ELECTROMAGNETIC COMPATIBILITY (EMC) IMPROVEMENTS

Primary attention was paid on the reduction of electromagnetic emission of the microcontroller P83CE654.

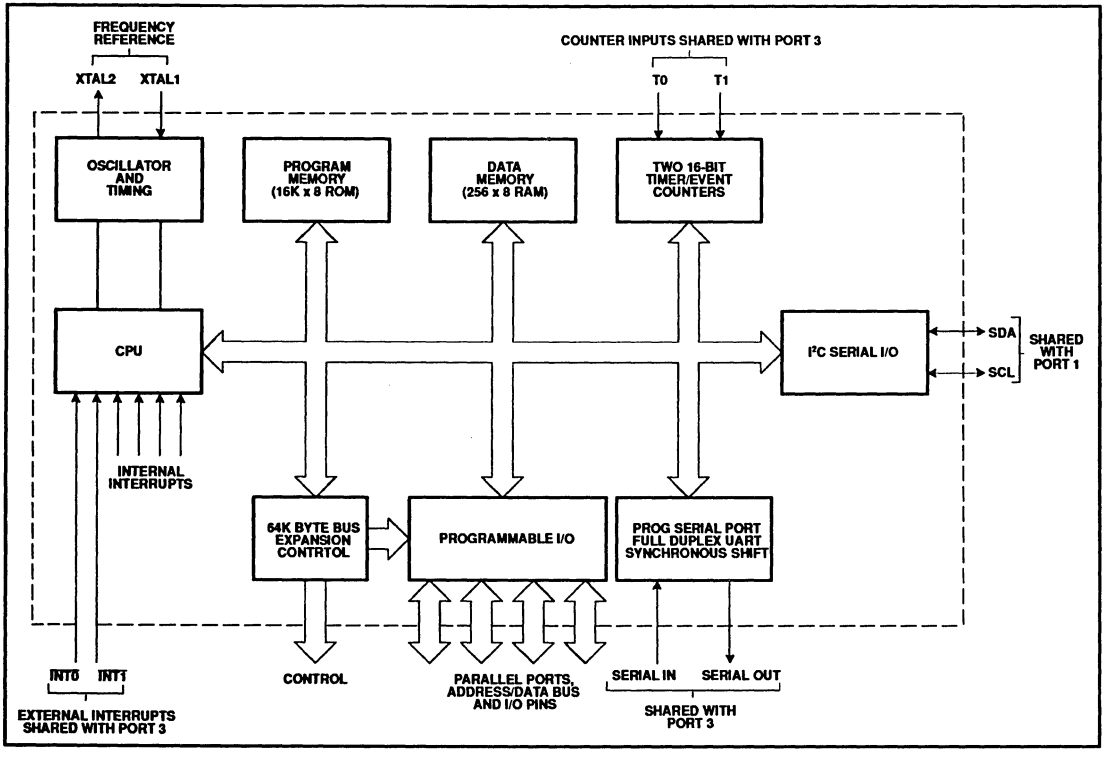
The following features effect in reducing the electromagnetic emission and additionally improve the electromagnetic susceptibility:

- Two supply voltage pins (V_{CC}) and four ground pins (V_{SS}) with a pair of V_{CC} and V_{SS} at two adjacent pins in the center at one side of the package and at two adjacent pins at the opposite side of the package and one more V_{SS} pin at each of the other two sides of the package
- Separate V_{CC} pins for the internal logic and the port buffers
- Internal decoupling capacitance improves the EMC radiation behavior and the EMC immunity
- External capacitors are to be located as close as possible between pins V_{CC2} and V_{SS3} as well as V_{CC1} and V_{SS2}; ceramic chip capacitors are recommended (100nF).

Useful in applications that require no external memory or temporarily no external memory:

- The ALE output signal (pulses at a frequency of $f_{OSC}/6$) can be disabled under software control (bit 5 in the SFR PCON: "RFI"); if disabled, no ALE pulse will occur. ALE pin will be pulled down internally, switching an external address latch to a quiet state. The MOVX instruction will still toggle ALE as a normal MOVX. ALE will retain its normal high value during Idle mode and a low value during Power-down mode while in the "RFI" reduction mode. Additionally during internal access ($\overline{EA}$ = 1) ALE will toggle normally when the address exceeds the internal program memory size. During external access ($\overline{EA}$ = 0) ALE will always toggle normally, whether the flag "RFI" is set or not.

BLOCK DIAGRAM



CMOS single-chip 8-bit microcontroller with Electromagnetic Compatibility improvements

PIN DESCRIPTIONS

MNEMONIC	PIN NUMBER	TYPE	NAME AND FUNCTION
V_{SS1}, V_{SS2}, V_{SS3}, V_{SS4}	16, 28, 39, 6	I	**Ground:** 0V reference. All pins must be connected.
V_{CC1}, V_{CC2}	17, 38	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation. Both pins must be connected.
P0.0–0.7	37–30	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s. Port 0 can sink/source 8 LSTTL inputs.
P1.0–P1.7	40–44, 1–3	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups, except P1.6 and P1.7 which are open drain. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 1 also receives the low-order address byte during program memory verification. Alternate functions include:
P1.6	2	I/O	**SCL:** I^2C-bus serial port clock line.
P1.7	3	I/O	**SDA:** I^2C-bus serial port data line.
P2.0–P2.7	18–25	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	5, 7–13	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 80C51 family, as listed below:
	5	I	**RxD (P3.0):** Serial input port
	7	O	**TxD (P3.1):** Serial output port
	8	I	**$\overline{INT0}$ (P3.2):** External interrupt 0 or gate control input for timer/event counter 0
	9	I	**$\overline{INT1}$ (P3.3):** External interrupt 1 or gate control input for timer/event counter 1
	10	I	**T0 (P3.4):** Timer 0 external input
	11	I	**T1 (P3.5):** Timer 1 external input
	12	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	13	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	4	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal pull-down resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE	27	I/O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. ALE can sink/sourse 8 LSTTL inputs. It can drive CMOS inputs without an external pull-up. To prohibit the toggling of ALE pin (RFI noise reduction) the bit RFI in the PCON Register (PCON.5) must be set by software. This bit is cleared on RESET and can be cleared by software. When set, ALE pin will be pulled down internally, switching an external address latch to a quiet state. The MOVX instruction will still toggle ALE as a normal MOVX. ALE will retain its normal high value during Idle mode and a low value during Power-down mode while in the "RFI" mode. Additionally during internal access ($\overline{EA}$ = 1) ALE will toggle normally when the address exceeds the internal program memory size. During external access ($\overline{EA}$ = 0) ALE will always toggle normally, whether the flag "RFI" is set or not.
$\overline{PSEN}$	26	O	**Program Store Enable:** The read strobe to external program memory. When the 8XCE654 is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory. $\overline{PSEN}$ can sink/source 8 LSTTL inputs.
$\overline{EA}$	29	I	**External Access Enable:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H and 3FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 3FFFH. $\overline{EA}$ is not allowed to float.
XTAL1	15	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	14	O	**Crystal 2:** Output from the inverting oscillator amplifier.

NOTE:
To avoid "latch-up" effect at power-on, the voltage on any pin at any time must not be higher or lower than V_{CC} + 0.5V or V_{SS} – 0.5V, respectively.

ROM CODE PROTECTION (83CE654)

The 83CE654 has an additional security feature. ROM code protection may be selected by setting a mask-programmable security bit (i.e., user dependent). This feature may be requested during ROM code submission. When selected, the ROM code is protected and cannot be read out at any time by any test mode or by any instruction in the external program memory space.

The MOVC instructions are the only instructions that have access to program code in the internal or external program memory. The $\overline{EA}$ input is latched during RESET and is "don't care" after RESET. This implementation prevents reading internal program code by switching from external program memory to internal program memory during a MOVC instruction or any other instruction that uses immediate data.

Table 1 lists the access to the internal and external program memory by the MOVC instructions when the security bit has been set to a logical "1":

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the Logic Symbol, page 560.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

Reset

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-on reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

Power-on Reset (See Figure 1.)

When V_{CC} is turned on, and provided its rise-time does not exceed 10ms, an automatic reset can be obtained by connecting the RST pin to V_{CC} via a 2.2µF capacitor. When the power is switched on, the voltage on the RST pin is equal to V_{CC} minus the capacitor voltage, and decreases from V_{CC} as the capacitor charges through the internal resistor (R_{RST}) to ground. The larger the capacitor, the more slowly V_{RST} decreases. V_{RST} must remain above the lower threshold of the Schmitt trigger long enough to effect a complete reset. The time required is the oscillator start-up time, plus 2 machine cycles.

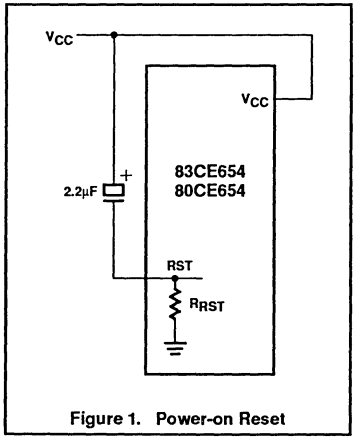


Figure 1. Power-on Reset

Idle Mode

In the idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

Power-Down Mode

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON. Table 2 shows the state of the I/O ports during low current operating modes.

Power Control Register PCON

These special modes are activated by software via the Special Function Register PCON. Its hardware address is 87H. PCON is not bit addressable. The reset value of PCON is (0x0x0000).

	7	6	5	4	3	2	1	0
PCON (87H)	SMOD	–	RFI	–	GF1	GF0	PD	IDL

Bit	Symbol	Function
PCON.7	SMOD	Double Baud rate bit. When set to logic 1 the baud rate is doubled when Timer 1 is used to generate baud rate, and the Serial Port is used in modes 1, 2 or 3.
PCON.6	–	(reserved for future use*)
PCON.5	RFI	When set to logic 1 the toggling of ALE pin is prohibited. This bit is cleared on RESET.
PCON.4	–	(reserved for future use*)
PCON.3	GF1	General purpose flag bit.
PCON.2	GF0	General purpose flag bit.
PCON.1	PD	Power-down bit. Setting this bit activates Power-down mode.
PCON.0	IDL	Idle mode bit. Setting this bit activates the Idle mode. If 1s are written to PD and IDL at the same time, PD takes precedence.

NOTE:

* User software should not write 1s to reserved bits. These bits may be used in future 80C51 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

Table 1.

	ACCESS TO INTERNAL PROGRAM MEMORY	ACCESS TO EXTERNAL PROGRAM MEMORY
MOVC in internal program memory	YES	YES
MOVC in external program memory	NO	YES

NOTE:
If the security bit has been set to a logical 0, there are no restrictions for the MOVC instructions.

Table 2. External Pin Status During Idle and Power-Down Mode

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Serial Control Register (S1CON) – See Table 3

S1CON (D8H)	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

Bits CR0, CR1 and CR2 determine the serial clock frequency that is generated in the master mode of operation.

Table 3. Serial Clock Rates

CR2	CR1	CR0	BIT FREQUENCY (kHz) AT f_{OSC} 6MHz	12MHz	16MHz	f_{OSC} DIVIDED BY
0	0	0	23	47	62.5	256
0	0	1	27	54	71	224
0	1	0	31.25	62.5	83.3	192
0	1	1	37	75	100	160
1	0	0	6.25	12.5	17	960
1	0	1	50	100	133[1]	120
1	1	0	100	200[1]	267[1]	60
1	1	1	0.25 < 31.25	0.5 < 62.5	0.67 < 56	96 × (256 – (reload value Timer 1)) (Reload value range: 0 – 254 in mode 2 for 6, 12MHz 0 – 253 in mode 2 for 16MHz)

NOTES:
1. These frequencies exceed the upper limit of 100kHz of the I^2C-bus specification and cannot be used in an I^2C-bus application.

ABSOLUTE MAXIMUM RATINGS

PARAMETER	RATING	UNIT
Voltage on V_{CC} to V_{SS}	–0.5 to +6.5	V
Voltage on any pin to V_{SS}	–0.5 to V_{CC}+0.5	V
Storage temperature range	–65 to +150	°C
Power dissipation (based on package heat transfer limitations, not device power consumption)[1]	1	W
Operating ambient temperature range		
FBB	0 to +70	°C
FFB	–40 to +85	°C

NOTE:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. this is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied. Exposure to Absolute Maximum Rating conditions for extended periods may affect device reliability.

DEVICE SPECIFICATIONS

TYPE	SUPPLY VOLTAGE (V)		FREQUENCY (MHz)		TEMPERATURE RANGE
	MIN.	MAX.	MIN.	MAX.	(°C)
P83(0)CE654FBB	4.5	5.5	1.2	16	0 to +70
P83(0)CE654FFB	4.5	5.5	1.2	16	–40 to +85

DC ELECTRICAL CHARACTERISTICS

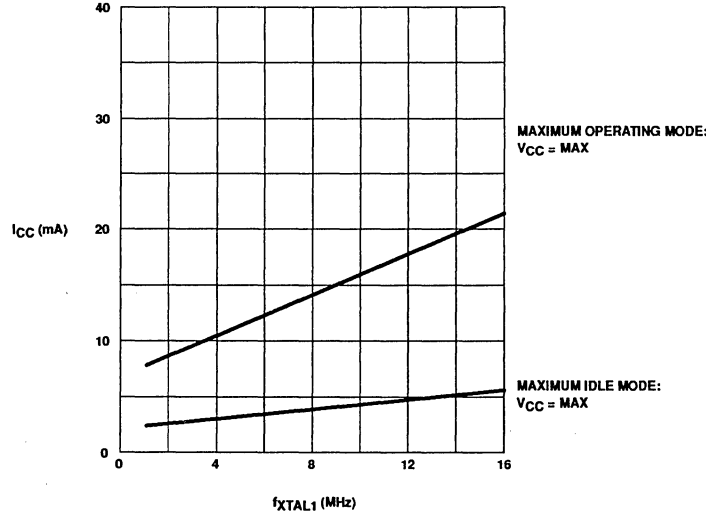
V_{CC} = 5V (±10%), V_{SS} = 0V, T_{amb} = 0°C to +70°C or –40°C to +85°C

SYMBOL	PARAMETER	PART TYPE	TEST CONDITIONS	LIMITS MIN.	LIMITS MAX.	UNIT
V_{IL}	Input low voltage, except $\overline{EA}$, P1.6/SCL, P1.7/SDA	0 to +70°C –40 to +85°C		–0.5 –0.5	$0.2V_{CC}$–0.1 $0.2V_{CC}$–0.15	V V
V_{IL1}	Input low voltage to $\overline{EA}$	0 to +70°C –40 to +85°C		–0.5 –0.5	$0.2V_{CC}$–0.3 $0.2V_{CC}$–0.35	V V
V_{IL2}	Input low voltage to P1.6/SCL, P1.7/SDA[6]			–0.5	$0.3V_{CC}$	V
V_{IH}	Input high voltage, except XTAL1, RST, P1.6/SCL, P1.7/SDA	0 to +70°C –40 to +85°C		$0.2V_{CC}$+0.9 $0.2V_{CC}$+1.0	V_{CC}+0.5 V_{CC}+0.5	V V
V_{IH1}	Input high voltage, XTAL1, RST	0 to +70°C –40 to +85°C		$0.7V_{CC}$ $0.7V_{CC}$+0.1	V_{CC}+0.5 V_{CC}+0.5	V V
V_{IH2}	Input high voltage, P1.6/SCL, P1.7/SDA[6]			$0.7V_{CC}$	6.0	V
V_{OL}	Output low voltage, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA[4]		I_{OL} = 1.6mA[7]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$[4]		I_{OL} = 3.2mA[7]		0.45	V
V_{OL2}	Output low voltage, P1.6/SCL, P1.7/SDA[4]		I_{OL} = 3.0mA[7]		0.4	V
V_{OH}	Output high voltage, ports 1, 2, 3, ALE, $\overline{PSEN}$		I_{OH} = –60µA I_{OH} = –25µA I_{OH} = –10µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$		V V V
V_{OH1}	Output high voltage; port 0 in external bus mode[5]		I_{OH} = –800µA I_{OH} = –300µA I_{OH} = –80µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$		V V V
I_{IL}	Logical 0 input current, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA	0 to +70°C –40 to +85°C	V_I = 0.45V V_I = 0.45V		–50 –75	µA µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3, except P1.6/SCL, P1.7/SDA	0 to +70°C –40 to +85°C	V_I = 2.0V V_I = 2.0V		–650 –750	µA µA
I_{L1}	Input leakage current, port 0, $\overline{EA}$		0.45V < V_I < V_{CC}		±10	µA
I_{L2}	Input leakage current, P1.6/SCL, P1.7/SDA		0V < V_I < 5.5V 0V < V_{CC} < 5.5V		±10	µA
I_{CC}	Power supply current: Active mode @ 16MHz[1, 8] Idle mode @ 16MHz[2, 8] Power down mode[3]		 V_{CC} = 5.5V V_{CC} = 5V ± 10% @2V < V_{PD} < V_{CCMAX}		 22 6 50	 mA mA µA
R_{RST}	Internal reset pull-down resistor			50	150	kΩ
C_{IO}	Pin capacitance		Freq.=1MHz		10	pF

NOTES: See Next Page.

NOTES FOR DC ELECTRICAL CHARACTERISTICS:

1. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 5ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{CC} - 0.5V$; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = P1.6 = P1.7 = V_{CC}.
2. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f = 5ns$; $V_{IL} = V_{SS} + 0.5V$; $V_{IH} = V_{CC} - 0.5V$; XTAL2 not connected; Port 0 = P1.6 = P1.7 = V_{CC}; $\overline{EA}$ = RST = V_{SS}.
3. The power-down current is measured with all output pins disconnected; XTAL2 not connected; Port 0 = P1.6 = P1.7 = V_{CC}; $\overline{EA}$ = XTAL1 = RST = V_{SS}.
4. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
5. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
6. The input threshold voltage of P1.6 and P1.7 (SIO1) meets the I^2C specification, so an input voltage below $0.3V_{CC}$ will be recognized as a logic 0 while an input voltage above $0.7V_{CC}$ will be recognized as a logic 1.
7. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows: Maximum I_{OL} = 10mA per port pin; Maximum I_{OL} = 26mA total for Port 0; Maximum I_{OL} = 15mA total for Ports 1, 2, and 3; Maximum I_{OL} = 71mA total for all output pins. If I_{OL} exceeds the test conditions, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
8. I_{CCMAX} for the 80/83CE654 at the other frequencies can be derived from Figure 1, where FREQ is the external oscillator frequency in MHz. I_{CCMAX} is given in mA.



VALID ONLY WITHIN FREQUENCY SPECIFICATIONS OF DEVICE UNDER TEST.

Figure 1. I_{CC} vs. Frequency

AC ELECTRICAL CHARACTERISTICS[1, 2]

SYMBOL	FIGURE	PARAMETER	16MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	2	Oscillator frequency			1.2	16	MHz
t_{LHLL}	2	ALE pulse width	85		$2t_{CLCL}-40$		ns
t_{AVLL}	2	Address valid to ALE low	8		$t_{CLCL}-55$		ns
t_{LLAX}	2	Address hold after ALE low	28		$t_{CLCL}-35$		ns
t_{LLIV}	2	ALE low to valid instruction in		150		$4t_{CLCL}-100$	ns
t_{LLPL}	2	ALE low to $\overline{PSEN}$ low	23		$t_{CLCL}-40$		ns
t_{PLPH}	2	$\overline{PSEN}$ pulse width	143		$3t_{CLCL}-45$		ns
t_{PLIV}	2	$\overline{PSEN}$ low to valid instruction in		83		$3t_{CLCL}-105$	ns
t_{PXIX}	2	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	2	Input instruction float after $\overline{PSEN}$		38		$t_{CLCL}-25$	ns
t_{AVIV}	2	Address to valid instruction in		208		$5t_{CLCL}-105$	ns
t_{PLAZ}	2	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{AVLL}	3, 4	Address valid to ALE low	28		$t_{CLCL}-35$		ns
t_{RLRH}	3, 4	$\overline{RD}$ pulse width	275		$6t_{CLCL}-100$		ns
t_{WLWH}	3, 4	$\overline{WR}$ pulse width	275		$6t_{CLCL}-100$		ns
t_{RLDV}	3, 4	$\overline{RD}$ low to valid data in		148		$5t_{CLCL}-165$	ns
t_{RHDX}	3, 4	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	3, 4	Data float after $\overline{RD}$		55		$2t_{CLCL}-70$	ns
t_{LLDV}	3, 4	ALE low to valid data in		350		$8t_{CLCL}-150$	ns
t_{AVDV}	3, 4	Address to valid data in		398		$9t_{CLCL}-165$	ns
t_{LLWL}	3, 4	ALE low to $\overline{RD}$ or $\overline{WR}$ low	138	238	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	3, 4	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	120		$4t_{CLCL}-130$		ns
t_{QVWX}	3, 4	Data valid to $\overline{WR}$ transition	3		$t_{CLCL}-60$		ns
t_{DW}	3, 4	Data setup time before $\overline{WR}$	288		$7t_{CLCL}-150$		ns
t_{WHQX}	3, 4	Data hold after $\overline{WR}$	13		$t_{CLCL}-50$		ns
t_{RLAZ}	3, 4	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	3, 4	$\overline{RD}$ or $\overline{WR}$ high to ALE high	23	103	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
Shift Register[3]							
t_{XLXL}	5	Serial port clock cycle time	0.75		$12t_{CLCL}$		µs
t_{QVXH}	5	Output data setup to clock rising edge	492		$10t_{CLCL}-133$		ns
t_{XHQX}	5	Output data hold after clock rising edge	80		$2t_{CLCL}-117$		ns
t_{XHDX}	5	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	5	Clock rising edge to input data valid		492		$10t_{CLCL}-133$	ns
External Clock							
t_{CHCX}	6	High time	20		20	$t_{CLCL}-t_{LOW}$	ns
t_{CLCX}	6	Low time	20		20	$t_{CLCL}-t_{HIGH}$	ns
t_{CLCH}	6	Rise time		20		20	ns
t_{CHCL}	6	Fall time		20		20	ns

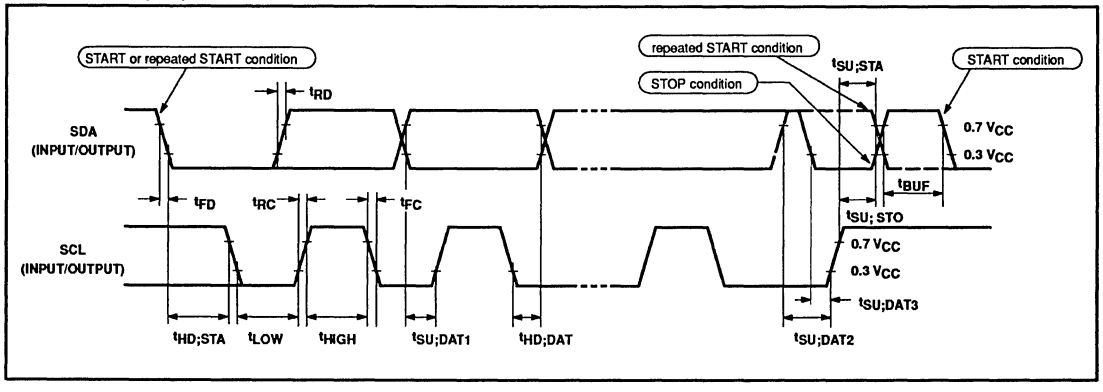
NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. Test condition: T_{amb} = 0°C to +70C; V_{CC} = 5V + 10%; V_{SS} = 0V; load capacitance = 80pF.

AC ELECTRICAL CHARACTERISTICS – I²C INTERFACE

SYMBOL	PARAMETER	INPUT	OUTPUT
SCL TIMING CHARACTERISTICS			
$t_{HD;STA}$	START condition hold time	≥ 14 t_{CLCL}	> 4.0µs[1]
t_{LOW}	SCL LOW time	≥ 16 t_{CLCL}	> 4.7µs[1]
t_{HIGH}	SCL HIGH time	≥ 14 t_{CLCL}	> 4.0µs[1]
t_{RC}	SCL rise time	≤ 1µs	–[2]
t_{FC}	SCL fall time	≤ 0.3µs	< 0.3µs[3]
SDA TIMING CHARACTERISTICS			
$t_{SU;DAT1}$	Data set-up time	≥ 250ns	> 20 t_{CLCL} – t_{RD}
$t_{SU;DAT2}$	SDA set-up time (before rep. START cond.)	≥ 250ns	> 1µs[1]
$t_{SU;DAT3}$	SDA set-up time (before STOP cond.)	≥ 250ns	> 8 t_{CLCL}
$t_{HD;DAT}$	Data hold time	≥ 0ns	> 8 t_{CLCL} – t_{FC}
$t_{SU;STA}$	Repeated START set-up time	≥ 14 t_{CLCL}	> 4.7µs[1]
$t_{SU;STO}$	STOP condition set-up time	≥ 14 t_{CLCL}	> 4.0µs[1]
t_{BUF}	Bus free time	≥ 14 t_{CLCL}	> 4.7µs[1]
t_{RD}	SDA rise time	≤ 1µs	–[2]
t_{FD}	SDA fall time	≤ 0.3µs	< 0.3µs[3]

NOTES:
1. At 100 kbit/s. At other bit rates this value is inversely proportional to the bit-rate of 100 kbit/s.
2. Determined by the external bus-line capacitance and the external bus-line pull-resistor, this must be < 1µs.
3. Spikes on the SDA and SCL lines with a duration of less than 3 t_{CLCL} will be filtered out. Maximum capacitance on bus-lines SDA and SCL = 400pF.
4. $t_{CLCL} = 1/f_{OSC}$ = one oscillator clock period at pin XTAL1. For 62ns < t_{CLCL} < 285ns (16MHz > f_{OSC} > 3.5MHz) the SIO1 interface meets the I²C-bus specification for bit-rates up to 100 kbit/s.

TIMING SIO1 (I²C) INTERFACE



Oscillator Circuitry

The capacitors connected to the crystal should be: C1 = C2 = 20pF.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

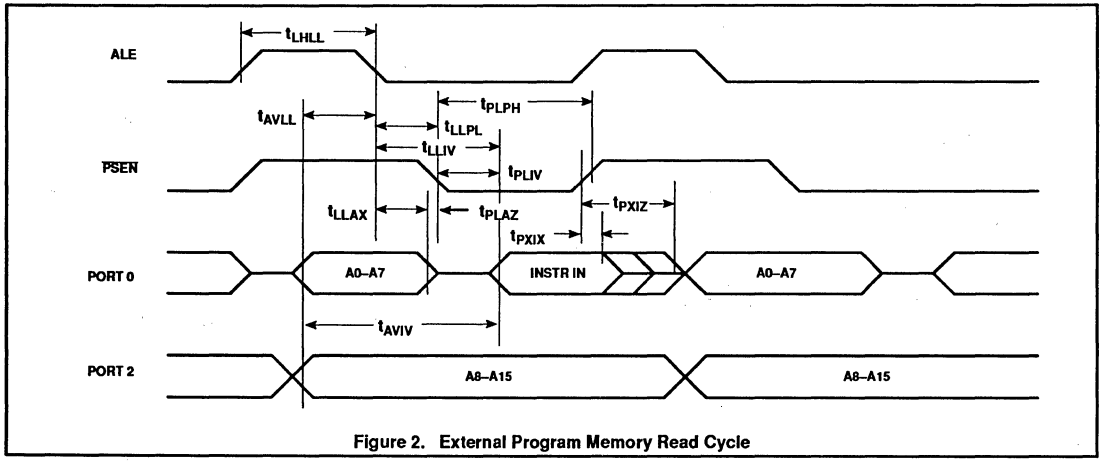


Figure 2. External Program Memory Read Cycle

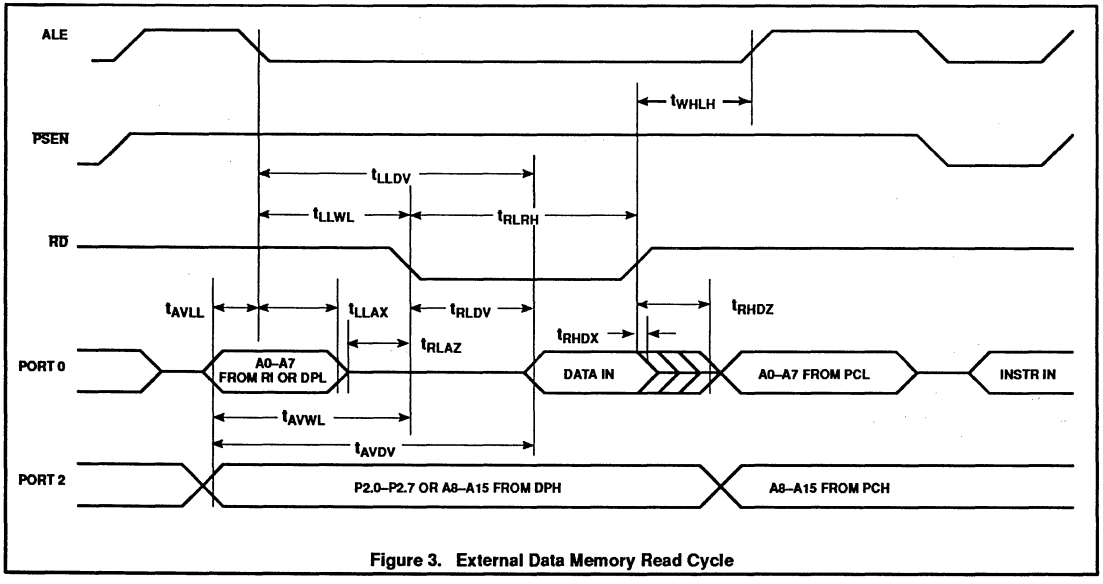


Figure 3. External Data Memory Read Cycle

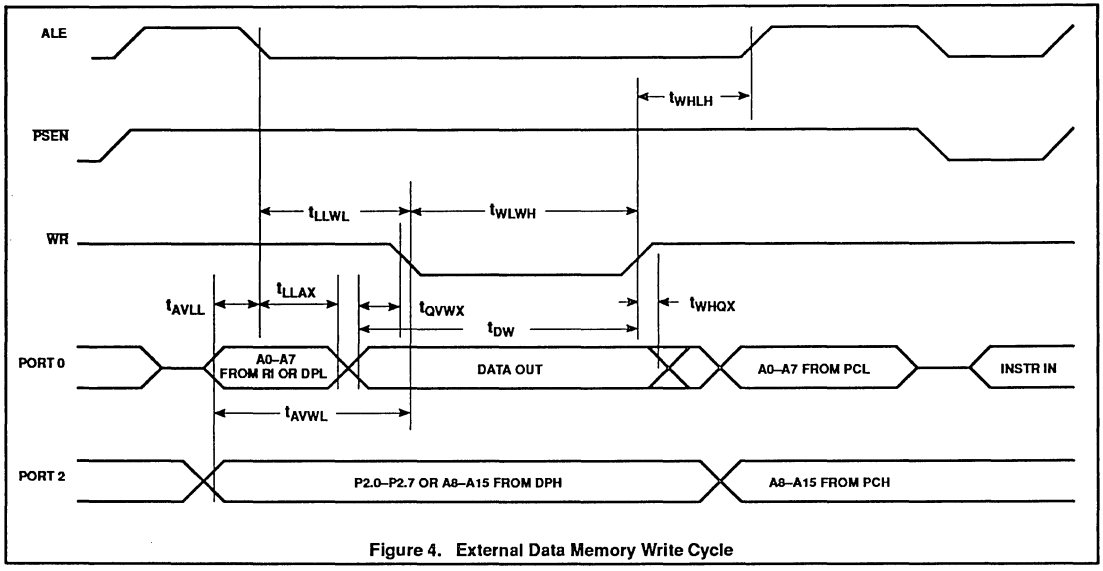


Figure 4. External Data Memory Write Cycle

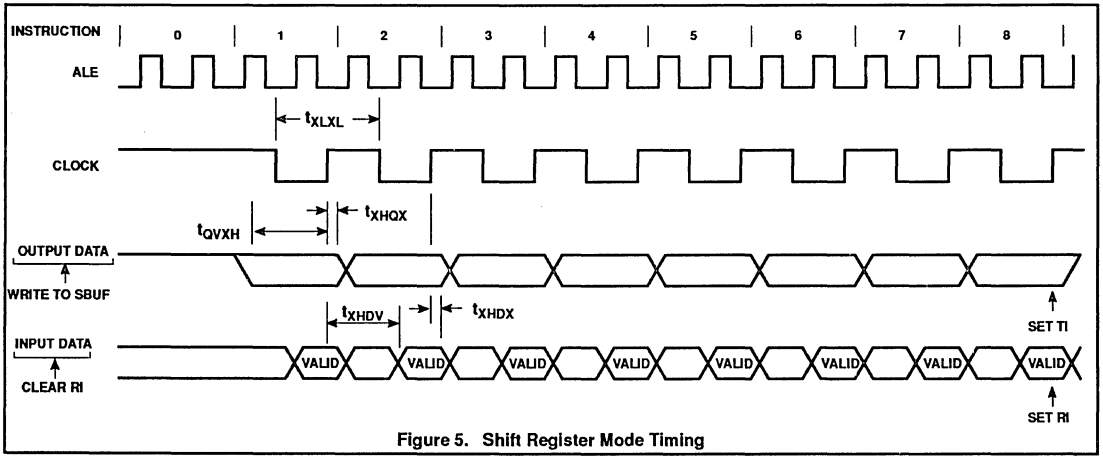


Figure 5. Shift Register Mode Timing

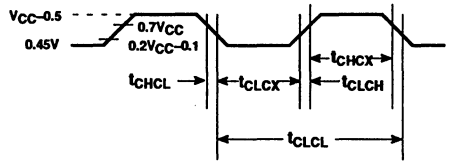
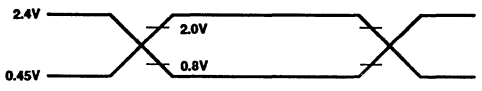
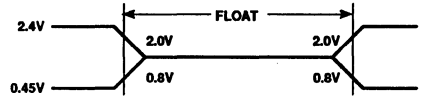


Figure 6. External Clock Drive



NOTE:
AC INPUTS DURING TESTING ARE DRIVEN AT 2.4V FOR A LOGIC '1' AND 0.45V FOR A LOGIC '0'. TIMING MEASUREMENTS ARE MADE AT 2.0V FOR A LOGIC '1' AND 0.8V FOR A LOGIC '0'.

Figure 7. AC Testing Input/Output



NOTE:
THE FLOAT STATE IS DEFINED AS THE POINT AT WHICH A PORT 0 PIN SINKS 3.2MA OR SOURCES 400μS AT THE VOLTAGE TEST LEVELS.

Figure 8. Float Waveform

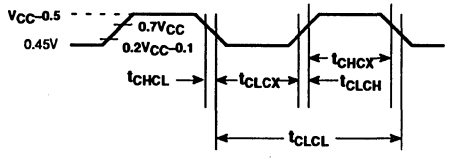


Figure 9. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specifications defined by Philips.

8XC751 OVERVIEW

The Signetics 83C751/87C751 offers the advantages of the SC80C51 architecture in a small package and at a low cost. This microcontroller is fabricated with Signetics high-density CMOS technology. Signetics epitaxial substrate minimizes CMOS latch-up sensitivity. The 83C751/87C751 (hereafter referred to collectively as the 83C751) contains a 2k × 8 ROM/EPROM, a 64 × 8 RAM, 19 I/O lines, a 16-bit auto-reload counter/timer, a fixed rate timer, a five source fixed priority interrupt structure, a bidirectional Inter-Integrated Circuit (I^2C) serial bus interface, and an on-chip oscillator. The onboard inter-integrated circuit (I^2C) bus interface allows the 83C751 to operate as a master or slave device on the I^2C small area network. This capability facilitates I/O and RAM expansion, access to EPROM, processor to processor communication, and efficient interface to a wide variety of dedicated I^2C peripherals. The 83C751 has the following features:

- SC80C51 based architecture
- Boolean processor
- Inter-integrated Circuit (I^2C) serial bus interface
- Fixed-rate timer
- 16-bit auto reloadable counter/timer
- Small package sizes
  - 24-pin DIP (300 mil "skinny DIP")
  - 28-pin PLCC
- 2k × 8 ROM/EPROM
- Available in erasable quartz lid (87C751), one-time programmable (87C751), or mask programmable versions (83C751)
- Wide oscillator frequency range
- Low power consumption:
  - –Normal operation: less than 11mA @ 5V, 12MHz
- Idle mode
- Power-down mode
- CMOS and TTL compatible

This part is well suited for logic replacement in consumer and industrial applications.

Differences from the 80C51

Instruction Set

PLEASE NOTE: The instruction set of the 83C751 is identical to the 80C51 except for the instructions: MOVX, LCALL, and LJUMP, which are not implemented. Care must be taken not to use any of these instructions in a user program, especially when using a high level language such as C.

Memory Organization

The central processing unit (CPU) manipulates operands in two address spaces as shown in Figure 1. The part's internal memory space consists of 2k bytes of program memory, and 64 bytes of data RAM overlapped with the 128-byte special function register area. The differences from the 80C51 are in RAM size (64 bytes vs. 128 bytes), in external RAM access (not available on the 83C751), in internal ROM size (2k bytes vs. 4k bytes), and in external program memory expansion (not available on the 83C751). The 128-byte special function register (SFR) space is accessed as on the 80C51 with some of the registers having been changed to reflect changes in the 83C751 peripheral functions. The stack may be located anywhere in internal RAM by loading the 8-bit stack pointer (SP). It should be noted that stack depth is limited to 64 bytes, the amount of available RAM. A reset loads the stack pointer with 07 (which is pre-incremented on a PUSH instruction).

Special Function Registers

The 83C751 contains many of the special function registers (SFR) that are found on the 80C51. Due to the different peripheral features on the 83C751, there are several additional SFRs and several that have been changed. There is no port 2 on the 83C751 so the P2 SFR isn't used. The standard UART found on the 80C51 has been replaced by the I^2C serial interface, so the UART SFRs, SCON, and SBUF have been replaced by I2CON and i2DAT, and two additional I^2C registers have been added (I2STA and I2CFG).

Because the interrupt structure is single level on the 83C751, there is no need for the IP SFR, so it is not used. The counter/timer has only one mode of operation, so the TMOD SFR is not used. There is also only one counter/timer, so there is no need for the TL1 and TH1 SFRs found on the 80C51. These have been replaced on the 83C751 by RTL and RTH, the counter/timer reload registers. Table 1 shows the special function registers, their locations, and reset values.

Data Pointer (DPTR)

The data pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). In the 80C51 this register allows the access of external data memory using the MOVX instruction. Since the 83C751 does not support MOVX or external memory accesses, this register is generally used as a 16-bit offset pointer of the accumulator in a MOVC instruction. DPTR may also be manipulated as two independent 8-bit registers.

I/O Port Latches (P0, P1, P3)

The port latches function the same as those on the 80C51. Since there is no port 2 on the 83C751, the P2 latch is not used. Port 0 on the 83C751 has only 3 bits, so only 3 bits of the P0 SFR have a useful function.

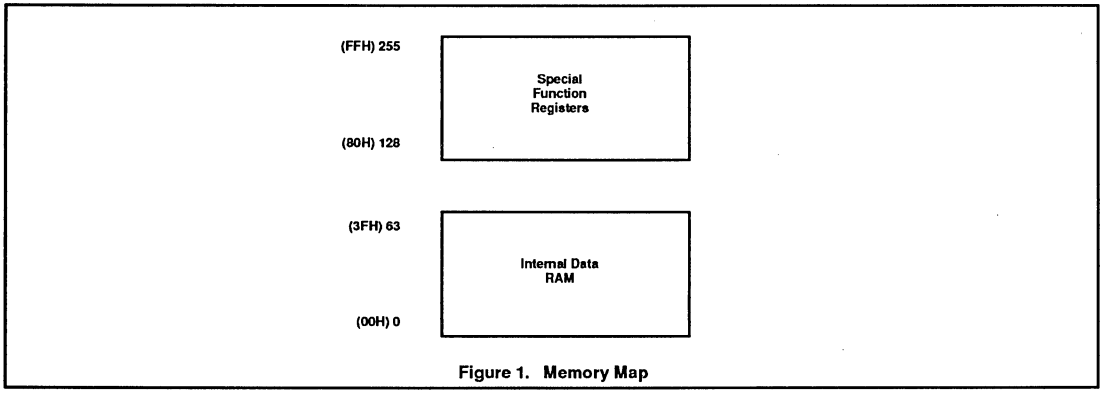


Figure 1. Memory Map

Table 1. 8XC751 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes)										
DPH	High byte	83H									00H
DPL	Low byte	82H									00H
			DF	DE	DD	DC	DB	DA	D9	D8	
I²CFG*#	I²C configuration	D8H/RD	SLAVEN	MASTRQ	0	TIRUN	–	–	CT1	CT0	0000xx00B
		WR	SLAVEN	MASTRQ	CLRTI	TIRUN	–	–	CT1	CT0	
			9F	9E	9D	9C	9B	9A	99	98	
I²CON*#	I²C control	98H/RD	RDAT	ATN	DRDY	ARL	STR	STP	MASTER	–	81H
		WR	CXA	IDLE	CDR	CARL	CSTR	CSTP	XSTR	XSTP	
I²DAT*#	I²C data	99H/RD	RDAT	0	0	0	0	0	0	0	80H
		WR	XDAT	X	X	X	X	X	X	X	
			FF	FE	FD	FC	FB	FA	F9	F8	
I²STA*#	I²C control	F8H	–	IDLE	XDATA	XACTV	MAKSTR	MAKSTP	XSTR	XSTP	x0100000B
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*#	Interrupt enable	A8H	EA	–	–	EI2	ETI	EX1	ET0	EX0	00H
								82	81	80	
P0*#	Port 0	80H	–	–	–	–	–	–	SDA	SCL	xxxxx111B
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	T0	$\overline{INT1}$	$\overline{INT0}$	–	–	–	–	–	FFH
P3*	Port 3	B0H	B7	B6	B5	B4	B3	B2	B1	B0	FFH
PCON#	Power control	87H	–	–	–	–	–	–	PD	IDL	xxxxxx00B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*#	Timer/counter control	88H	GATE	C/T	TF	TR	IE0	IT0	IE1	IT1	00H
TI#	Timer low byte	8AH									00H
TH#	Timer high byte	8CH									00H
RTL#	Timer low reload	8BH									00H
RTH#	Timer high reload	8DH									00H

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.

I/O Port Structure

The 8XC751 has two 8-bit ports (ports 1 and 3) and one 3-bit port (port 0). All three ports on the 8XC751 are bidirectional. Each consists of a latch (special function register P0, P1, P3), an output driver, and an input buffer. Three port 1 pins and two port 0 pins are multifunctional. In addition to being port pins, these pins serve the function of special features as follows:

Port Pin	Alternate Function
P0.0	I^2C clock (SCL)
P0.1	I^2C data (SDA)
P1.5	INT0 (external interrupt 0 input)
P1.6	INT1 (external interrupt 1 input)
P1.7	T0 (timer 0 external input)

Ports 1 and 3 are identical in structure to the same ports on the 80C51. The structure of port 0 on the 8XC751 is similar to that of the 80C51 but does not include address/data input and output circuitry. As on the 80C51, ports 1 and 3 are quasi-bidirectional while port 0 is bidirectional with no internal pullups.

Timer/Counter

The 8XC751 has two timers: a 16-bit timer/counter and a 10-bit fixed-rate timer. The 16-bit timer/counter's operation is similar to mode 2 operation on the 80C51, but is extended to 16 bits. The timer/counter is clocked by either 1/12 the oscillator frequency or by transitions on the T0 pin. The C/T pin in special function register TCON selects between these two modes. When the TCON TR bit is set, the timer/counter is enabled. Register pair TH and TL are incremented by the clock source. When the register pair overflows, the register pair is reloaded with the values in registers RTH and RTL. The value in the reload registers is left unchanged. See the 83C751 counter/timer block diagram in Figure 2. The TF bit in special function register TCON is set on counter overflow and, if the interrupt is enabled, will generate an interrupt.

TCON Register

GATE	C/T	TF	TR	IE0	IT0	IE1	IT1
MSB							LSB

GATE 1 – Timer/counter is enabled only when INT0 pin is high, and TR is 1.
0 – Timer/counter is enabled whenTR is 1.

C/T 1 – Counter/timer operation from T0 pin.
0 – Timer operation from internal clock.

TF 1 – Set on overflow of TH.
0 – Cleared when processor vectors to interrupt routine and by reset.

TR 1 – Timer/counter enabled.
0 – Timer/counter disabled.

IE0 1 – Edge detected in $\overline{INT0}$.

IT0 1 – $\overline{INT0}$ is edge triggered.
0 – $\overline{INT0}$ is level sensitive.

IE1 1 – Edge detected on $\overline{INT1}$.

IT1 1 – $\overline{INT1}$ is edge triggered.
0 – $\overline{INT1}$ is level sensitive.

These flags are functionally identical to the corresponding 80C51 flags, except that there is only one timer on the 83C751 and the flags are therefore combined into one register.

Note that the positions of the IE0/IT0 and IE1/IT1 bits are transposed from the positions used in the standard 80C51 TCON register.

Timer I is used to control the timing of the I^2C bus and also to detect a "bus locked" condition, by causing an interrupt when nothing happens on the I^2C bus for an inordinately long period of time while a transmission is in progress. If the interrupt does not occur, the program can attempt to correct the fault and allow the last I^2C transmission to be repeated.

The I^2C watchdog timer, timer I, is also available as a general-purpose fixed-rate timer when the I^2C interface is not being used. A clock rate of 1/12 the oscillator frequency forms the input to the timer. Timer I has a timeout interval of 1024 machine cycles when used as a fixed-rate timer.

I^2C Serial Interface

The I^2C bus uses two wires (SDA and SCL) to transfer information between devices connected to the bus. The main features of the bus are:

- Bidirectional data transfer between masters and slaves
- Serial addressing of slaves (no added wiring)
- Acknowledgment after each transferred byte
- Multimaster bus
- Arbitration between simultaneously transmitting masters without corruption of serial data on bus

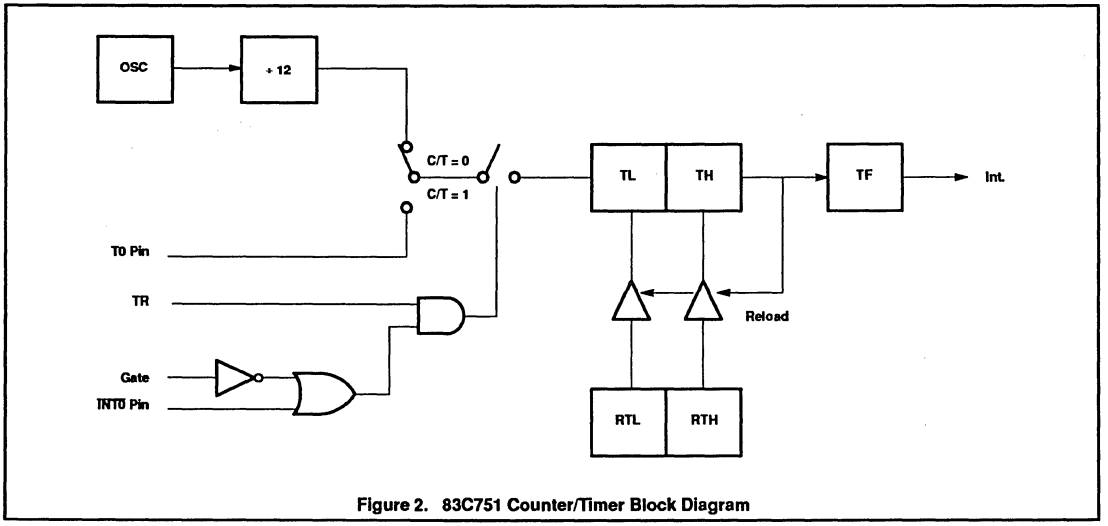


Figure 2. 83C751 Counter/Timer Block Diagram

A large family of I²C compatible ICs is available. See the I²C section of this manual for more details on the bus and available ICs.

The 83C751 I²C subsystem includes hardware to simplify the software required to drive the I²C bus. The hardware is a single bit interface which in addition to including the necessary arbitration and framing error checks, includes clock stretching and a bus timeout timer. The interface is synchronized to software either through polled loops or interrupts. Refer to the application note AN422, in Section 4, entitled "Using the 8XC751 Microcontroller as an I²C Bus Master" for additional discussion of the 83C751 I²C interface and sample driver routines.

Six time spans are important in I²C operation and are insured by timer I:

- The MINIMUM HIGH time for SCL when this device is the master.
- The MINIMUM LOW time for SCL when this device is a master. This is not very important for a single-bit hardware interface like this one, because the SCL low time is stretched until the software responds to the I²C flags. The software response time normally meets or exceeds the MIN LO time. In cases where the software responds within MIN HI + MIN LO) time, timer I will ensure that the minimum time is met.
- The MINIMUM SCL HIGH TO SDA HIGH time in a stop condition.
- The MINIMUM SDA HIGH TO SDA LOW time between I²C stop and start conditions (4.7µs, see spec.).
- The MINIMUM SDA LOW TO SCL LOW time in a start condition.
- The MAXIMUM SCL CHANGE time while an I²C frame is in progress. A frame is in progress between a start condition and the following stop condition. This time span serves to detect a lack of software response on this 8XC751 as well as external I²C problems. SCL "stuck low" indicates a faulty master or slave. SCL "stuck high" may mean a faulty device, or that noise induced onto the I²C bus caused all masters to withdraw from I²C arbitration.

The first five of these times are 4.7µs (see I²C specification) and are covered by the low order three bits of timer I. Timer I is clocked by the 8XC751 oscillator, which can vary in frequency from 0.5 to 16MHz. Timer I can be preloaded with one of four values to optimize timing for different oscillator frequencies. At lower frequencies, software response time is increased and will degrade maximum performance of the I²C bus. See special function register I2CFG description for prescale values (CT0, CT1).

The MAXIMUM SCL CHANGE time is important, but its exact span is not critical. The complete 10 bits of timer I are used to count out the maximum time. When I²C operation is enabled, this counter is cleared by transitions on the SCL pin. The timer does not run between I²C frames (i.e., whenever reset or stop occurred more recently than the last start). When this counter is running, it will carry out after 1020 to 1023 machine cycles have elapsed since a change on SCL. A carry out causes a hardware reset of the 83C751 I²C interface and generates an interrupt if the timer I interrupt is enabled. In cases where the bus hangup is due to a lack of software response by this 83C751, the reset releases SCL and allows I²C operation among other devices to continue.

I²C Interrupts

If I²C interrupts are enabled (EA and EI2 are both set to 1), an I²C interrupt will occur whenever the ATN flag is set by a start, stop, arbitration loss, or data ready condition (refer to the description of ATN following). In practice, it is not efficient to operate the I²C interface in this fashion because the I²C interrupt service routine would somehow have to distinguish between hundreds of possible conditions. Also, since I²C can operate at a fairly high rate, the software may execute faster if the code simply waits for the I²C interface.

Typically, the I²C interrupt should only be used to indicate a start condition at an idle slave device, or a stop condition at an idle master device (if it is waiting to use the I²C bus). This is accomplished by enabling the I²C interrupt only during the aforementioned conditions.

I²C Register I2CON

	7	6	5	4	3	2	1	0
Read	RDAT	ATN	DRDY	ARL	STR	STP	MASTER	–
Write	CXA	IDLE	CDR	CARL	CSTR	CSTP	XSTR	XSTP

Reading I2CON

RDAT The data from SDA is captured into "Receive DATa" whenever a rising edge occurs on SCL. RDAT is also available (with seven low-order zeros) in the I2DAT register. The difference between reading it here and there is that reading I2DAT clears DRDY, allowing the I²C to proceed on to another bit. Typically, the first seven bits of a received byte are read from I2DAT, while the 8th is read here. Then I2DAT can be written to send the Ack bit and clear DRDY.

ATN "ATteNtion" is 1 when one or more of DRDY, ARL, STR, or STP is 1. Thus, ATN comprises a single bit that can be tested to release the I²C service routine from a "wait loop."

DRDY "Data ReaDY" (and thus ATN) is set when a rising edge occurs on SCL, except at idle slave. DRDY is cleared by writing CDR = 1, or by writing or reading the I2DAT register. The following low period on SCL is stretched until the program responds by clearing DRDY.

Checking ATN and DRDY

When a program detects ATN = 1, it should next check DRDY. If DRDY = 1, then if it receives the last bit, it should capture the data from RDAT (in I2DAT or I2CON). Next, if the next bit is to be sent, it should be written to I2DAT. One way or another, it should clear DRDY and then return to monitoring ATN. Note that if any of ARL, STR, or STP is set, clearing DRDY will not release SCL to high, so that the I²C will not go on to the next bit. If a program detects ATN = 1, and DRDY = 0, it should go on to examine ARL, STR, and STP.

ARL "Arbitration Loss" is 1 when transmit Active was set, but this 83C751 lost arbitration to another transmitter. Transmit Active is cleared when ARL is 1. There are four separate cases in which ARL is set..

1. If the program sent a 1 or repeated start, but another device sent a 0, or a stop, so that SDA is 0 at the rising edge of SCL. (If the other device sent a stop, the setting of ARL will be followed shortly by STP being set.)
2. If the program sent a 1, but another device sent a repeated start, and it drove SDA low before the 83C751 could drive SCL low. (This type of ARL is always accompanied by STR = 1.)
3. In master mode, if the program sent a repeated start, but another device sent a 1, and it drove SCL low before this 83C751 could drive SDA low.
4. In master mode, if the program sent stop, but it could not be sent because another device sent a 0.

STR "STaRt" is set to a 1 when an I²C start condition is detected at a non-idle slave or at a master. (STR is not set when an idle slave becomes active due to a start bit; the slave has nothing useful to do until the rising edge of SCL sets DRDY.)

STP "SToP" is set to 1 when an I²C stop condition is detected at a non-idle slave or at a master. (STP is not set for a stop condition at an idle slave.)

MASTER "MASTER" is 1 if this 83C751 is currently a master on the I²C. MASTER is set when MASTRQ is 1 and the bus is not busy (i.e., if a start bit hasn't been received since reset or a "Timer I" time-out, or if a stop has been received since the last start). MASTER is cleared when ARL is set, or after the software writes MASTRQ = 0 and then XSTP = 1.

Writing I2CON

Typically, for each bit in an I²C message, a service routine waits for ATN = 1. Based on DRDY, ARL, STR, and STP, and on the current bit position in the message, it may then write I2CON with one or more of the following bits, or it may read or write the I2DAT register.

CXA Writing a 1 to "Clear Xmit Active" clears the Transmit Active state. (Reading the I2DAT register also does this.)

Regarding Transmit Active

Transmit Active is set by writing the I2DAT register, or by writing I2CON with XSTR = 1 or XSTP = 1. The I²C interface will only drive the SDA line low when Transmit Active is set, and the ARL bit will only be set to 1 when Transmit Active is set. Transmit Active is cleared by reading the I2DAT register, or by writing I2CON with CXA = 1. Transmit Active is automatically cleared when ARL is 1.

IDLE Writing 1 to "IDLE" causes a slave's I²C hardware to ignore the I²C until the next start condition (but if MASTRQ is 1, then a stop condition will make the 83C751 into a master).

CDR Writing a 1 to "Clear Data Ready" clears DRDY. (Reading or writing the I2DAT register also does this.)

CARL Writing a 1 to "Clear Arbitration Loss" clears the ARL bit.

CSTR Writing a 1 to "Clear STaRt" clears the STR bit.

CSTP Writing a 1 to "Clear SToP" clears the STP bit. Note that if one or more of DRDY, ARL, STR, or STP is 1, the low time of SCL is stretched until the service routine responds by clearing them.

XSTR Writing 1s to "Xmit repeated STaRt" and CDR tells the I²C hardware to send a repeated start condition. This should only be at a master. Note that XSTR need not and should not be used to send an "initial" (nonrepeated) start; it is sent automatically by the I²C hardware. Writing XSTR = 1 includes the effect of writing I2DAT with XDAT = 1; it sets Transmit Active and releases SDA to high during the SCL low time. After SCL goes high, the I²C hardware waits for the suitable minimum time and then drives SDA low to make the start condition.

XSTP Writing 1s to "Xmit SToP" and CDR tells the I²C hardware to send a stop condition. This should only be done at a master. If there are no more messages to initiate, the service routine should clear the MASTRQ bit in I2CFG to 0 before writing XSTP with 1. Writing XSTP = 1 includes the effect of writing I2DAT with XDAT = 0; it sets Transmit Active and drives SDA low during the SCL low time. After SCL goes high, the I²C hardware waits for the suitable minimum time and then releases SDA to high to make the stop condition.

I²C Register I2DAT

	7	6	5	4	3	2	1	0
Read	RDAT	0	0	0	0	0	0	0
Write	XDAT	X	X	X	X	X	X	X

RDAT "Receive DATa" is captured from SDA every rising edge of SCL. Reading I2DAT also clears DRDY and the Transmit Active state.

XDAT "Xmit Data" sets the data for the next bit. Writing I2DAT also clears DRDY and sets the Transmit Active state.

Regarding Software Response Time

Because the 83C751 can run at 16MHz, and because the I²C interface is optimized for high-speed operation, it is quite likely that an I²C service routine will sometimes respond to DRDY (which is set at a rising edge of SCL) and write I2DAT before SCL has gone low again. If XDAT were applied directly to SDA, this situation would produce an I²C protocol violation. The programmer need not worry about this possibility because XDAT is applied to SDA only when SCL is low.

Conversely, a program that includes an I²C service routine may take a long time to respond to DRDY. Typically, an I²C routine operates on a flag-polling basis during a message, with interrupts from other peripheral functions enabled. If an interrupt occurs, it will delay the response of the I²C service routine. The programmer need not worry about this very much either, because the I²C hardware stretches the SCL low time until the service routine responds. The only constraint on the response is that it must not exceed the Timer I time-out, which is at least 765 microseconds.

I²C Register I2CFG

	7	6	5	4	3	2	1	0
Read	SLAVEN	MASTRQ	0	TIRUN	–	–	CT1	CT0
Write	SLAVEN	MASTRQ	CLRTI	TIRUN	–	–	CT1	CT0

SLAVEN Writing a 1 to "SLAVe ENable" enables the slave functions of the I²C subsystem. If SLAVEN and MASTRQ are 0, the I²C hardware is disabled. This bit is cleared to 0 by reset and by an I²C time-out.

MASTRQ Writing a 1 to "MASTRQ" requests mastership of the I²C. If a frame from another master is in progress when this bit is changed from 0 to 1, action is delayed until a stop condition is detected. Then, or immediately if a frame is not in progress, a start condition is sent and DRDY is set (thus making ATN 1 and generating an I²C interrupt). When a master wishes to release mastership status of the I²C, it writes a 1 to XSTP in I2CON. MASTRQ is cleared by reset and by an I²C time-out.

CLRTI Writing a 1 to this bit clears the Timer I interrupt flag. This bit position always reads as a 0.

TIRUN Writing a 1 to this bit lets Timer I run; a zero stops and clears it. Together with SLAVEN, MASTRQ, and MASTER, this bit determines operational modes as shown in Table 2.

CT1,0 These two bits are programmed as a function of the OSC rate, to optimize the MIN HI and LO time of SCL when this 83C751 is a master on the I²C. The time value determined by these bits controls both of these parameters, and also the timing for stop and start conditions. These bits are cleared to 00 by reset.

Table 2. Interaction of TIRUN with SLAVEN, MASTRQ, and MASTER

SLAVEN, MASTRQ, MASTER	TIRUN	OPERATING MODE
All 0	0	The I²C interface is disabled. Timer I is cleared and does not run. This is the state assumed after a reset. If an I²C application wants to ignore the I²C at certain times, it should write SLAVEN, MASTRQ, and TIRUN all to zero.
All 0	1	The I²C interface is disabled. Timer I operates as a free-running time base. Use this mode only in non-I²C applications.
Any or all 1	0	The I²C interface is enabled. The 3 low-order bits of Timer I run for min-time generation, but the hi-order bits do not, so that there is no checking for I²C being "hung." This configuration can be used for very slow I²C operation.
Any or all 1	1	The I²C interface is enabled. Timer I runs during frames on the I²C, and is cleared by transitions on SCL, and by Start and Stop conditions. This is the normal state for I²C operation.

Values to be used in the CT1 and CT0 bits are shown in Table 3. To allow the I²C bus to run at the maximum rate for a particular oscillator frequency, compare the actual oscillator rate to the f_{OSC} max column in the table. The value for CT1 and CT0 is found in the first line of the table where f_{OSC} max is greater than or equal to the actual frequency.

The table also shows the osc/12 count for various settings of CT1/CT0. This allows calculation of the actual minimum high and low times for SCL as follows:

$$\text{SCL min high/low time} = \frac{12 * \text{count}}{\text{osc (in MHz)}} \quad \text{(in microseconds)}$$

For instance, at a 16MHz frequency, with CT1/CT0 set to 10, the minimum SCL high and low times will be 5.25µs

The table also shows the Timer I timeout period (given in machine cycles) for each CT1/CT0 combination. The timeout period varies because of the way in which minimum SCL high and low times are measured. When the I²C interface is operating, Timer I is preloaded at every SCL transition with a value dependent upon CT1/CT0. The preload value is chosen such that a minimum SCL high or low time has elapsed when Timer I reaches a count of 008 (the actual value preloaded into Timer I is 8 minus the osc/12 count).

I²C Register I2STA

Read only

7	6	5	4	3	2	1	0
–	IDLE	XDATA	XACTV	MAKSTR	MAKSTP	XSTR	XSTP
MSB							LSB

This register is read only and reflects the internal status of the I²C hardware. IDLE, XSTR, and XSTP reflect the status of the like named bits in the I2CON register.

XDATA The content of the transmitter buffer.

XACTV Transmitter active.

MAKSTR This bit is high while the hardware is effecting a start condition.

MAKSTP This bit is high while the hardware is effecting a stop condition.

XSTR This bit is active while the hardware is effecting a repeated start condition.

XSTP This bit is active while the hardware is effecting a repeated stop condition.

Interrupts

The interrupt structure is a five-source, one-level interrupt system. Interrupt sources common to the 80C51 are the external interrupts ($\overline{INT0}$, $\overline{INT1}$) and the timer/counter interrupt (ET0). The I²C interrupt (EI2) and Timer I interrupt (ETI) are the other two interrupt sources. The interrupt sources are listed below in their order of polling sequence priority.

Upon interrupt or reset the program counter is loaded with specific values for the appropriate interrupt service routine in program memory. These values are:

Event	Program Memory Address	Priority
Reset	000	Highest
INT0	003	
Counter/Timer 0	00B	
INT1	013	
Timer I	01B	
I²C	023	Lowest

The interrupt enable register (IE) is used to individually enable or disable the five sources. Bit $\overline{EA}$ in the interrupt enable register can be used to globally enable or disable all interrupt sources. The interrupt enable register is described below. All other interrupt details are based on the 80C51 interrupt architecture.

Table 3. CT1, CT0 Values

CT1, CT0	OSC/12 COUNT	f_{OSC} MAX	TIMEOUT PERIOD
10	7	16.8MHz	1023 cycles
01	6	14.25MHz	1022 cycles
00	5	11.7MHz	1021 cycles
11	4	9.14MHz	1020 cycles

Interrupt Enable Register

$\overline{EA}$	X	X	EI2	ETI	EX1	ET0	EX0

Symbol	Position	Function
$\overline{EA}$	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit
–	IE.6	Reserved
–	IE.5	Reserved
EI2	IE.4	Enables or disables the I^2C interrupt. If EI2 = 0, the I^2C interrupt is disabled
ETI	IE.3	Enables or disables the Timer I overflow interrupt. If ETI = 0, the Timer I interrupt is disabled.
EX1	IE.2	Enables or disables external interrupt 1. If EX1 = 0, external interrupt 1 is disabled.
ET0	IE.1	Enables or disables the Timer 0 overflow interrupt. If ET0 = 0, theTimer 0 interrupt is disabled.
EX0	IE.0	Enables or disables external interrupt 0. If EX0 = 0, external interrupt 0 is disabled.

CMOS single-chip 8-bit microcontroller — 83C751/87C751

DESCRIPTION

The Philips 83C751/87C751 offers the advantages of the 80C51 architecture in a small package and at low cost.

The 8XC751 Microcontroller is fabricated with Philips high-density CMOS technology. Philips epitaxial substrate minimizes CMOS latch-up sensitivity.

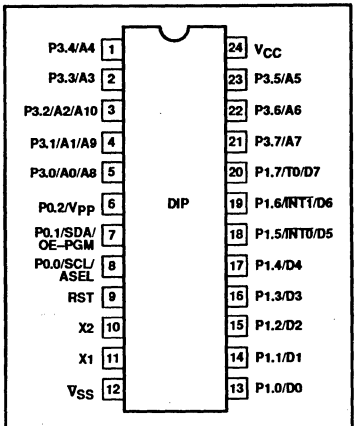
The 8XC751 contains a 2k × 8 ROM (83C751) EPROM (87C751), a 64 × 8 RAM, 19 I/O lines, a 16-bit auto-reload counter/timer, a five-source, fixed-priority level interrupt structure, a bidirectional inter-integrated circuit (I^2C) serial bus interface, and an on-chip oscillator.

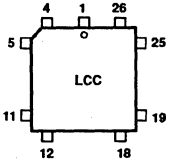
The on-board inter-integrated circuit (I^2C) bus interface allows the 8XC751 to operate as a master or slave device on the I^2C small area network. This capability facilitates I/O and RAM expansion, access to EEPROM, processor-to-processor communication, and efficient interface to a wide variety of dedicated I^2C peripherals.

FEATURES

- 80C51 based architecture
- Inter-Integrated Circuit (I^2C) serial bus interface
- Small package sizes
  - 24-pin DIP (300 mil "skinny DIP")
  - 28-pin PLCC
- 87C751 available in erasable quartz lid or one-time programmable plastic packages
- Wide oscillator frequency range
- Low power consumption:
  - Normal operation: less than 11mA @ 5V, 12MHz
  - Idle mode
  - Power-down mode
- 2k × 8 ROM (83C751)
  2k × 8 EPROM (87C751)
- 64 × 8 RAM
- 16-bit auto reloadable counter/timer
- Fixed-rate timer
- Boolean processor
- CMOS and TTL compatible
- Well suited for logic replacement, consumer and industrial applications

PIN CONFIGURATIONS



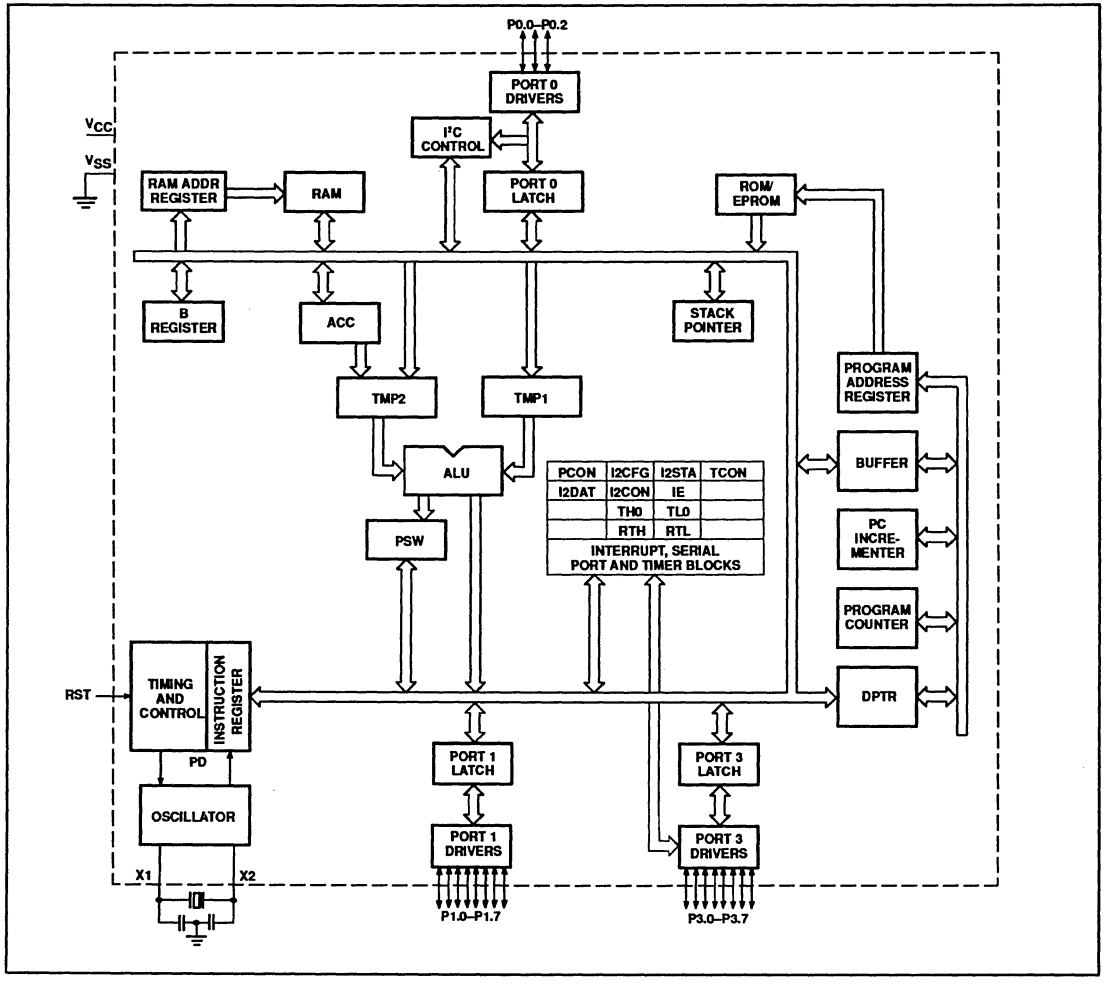


Pin	Function	Pin	Function
1	P3.4/A4	15	P1.0/D0
2	P3.3/A3	16	P1.1/D1
3	P3.2/A2/A10	17	P1.2/D2
4	P3.1/A1/A9	18	P1.3/D3
5	N.C.	19	P1.4/D4
6	P3.0/A0/A8	20	P1.5/INT0/D5
7	P0.2/V_{PP}	21	N.C.
8	P0.1/SDA/OE–PGM	22	N.C.
9	P0.0/SCL/ASEL	23	P1.6/INT1/D6
10	N.C.	24	P1.7/T0/D7
11	RST	25	P3.7/A7
12	X2	26	P3.6/A6
13	X1	27	P3.5/A5
14	V_{SS}	28	V_{CC}

PART NUMBER SELECTION

ROM	EPROM	SPEED	TEMPERATURE AND PACKAGE
	S87C751–1F24	3.5 to 12MHz	0 to +70°C, ceramic DIP
	S87C751–2F24	3.5 to 12MHz	–40 to +85°C, ceramic DIP
	S87C751–4F24	3.5 to 16MHz	0 to +70°C, ceramic DIP
	S87C751–5F24	3.5 to 16MHz	–40 to +85°C, ceramic DIP
S83C751–1N24	S87C751–1N24	3.5 to 12MHz	0 to +70°C, plastic DIP
S83C751–2N24	S87C751–2N24	3.5 to 12MHz	–40 to +85°C, plastic DIP
S83C751–4N24	S87C751–4N24	3.5 to 16MHz	0 to +70°C, plastic DIP
S83C751–5N24	S87C751–5N24	3.5 to 16MHz	–40 to +85°C, plastic DIP
S83C751–1A28	S87C751–1A28	3.5 to 12MHz	0 to +70°C, plastic LCC
S83C751–2A28	S87C751–2A28	3.5 to 12MHz	–40 to +85°C, plastic LCC
S83C751–4A28	S87C751–4A28	3.5 to 16MHz	0 to +70°C, plastic LCC
S83C751–5A28	S87C751–5A28	3.5 to 16MHz	–40 to +85°C, plastic LCC

BLOCK DIAGRAM



PIN DESCRIPTIONS

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	12	14	I	**Circuit Ground Potential**
V_{CC}	24	28	I	**Supply voltage during normal, Idle, and power-down operation.**
P0.0–P0.2	8–6	9–7	I/O	**Port 0:** Port 0 is a 3-bit open-drain, bidirectional port. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs. Port 0 also serves as the serial I^2C interface. When this feature is activated by software, SCL and SDA are driven low in accordance with the I^2C protocol. These pins are driven low if the port register bit is written with a 0 or if the I^2C subsystem presents a 0. The state of the pin can always be read from the port register by the program. To comply with the I^2C specification, P0.0 and P0.1 are open drain bidirectional I/O pins with the electrical characteristics listed in the tables that follow. While these differ from "standard TTL" characteristics, they are close enough for the pins to still be used as general-purpose I/O in non-I^2C applications. Port 0 also provides alternate functions for programming the EPROM memory as follows:
	6	7	N/A	**V_{PP} (P0.2)** – Programming voltage input.
	7	8	I	**OE/PGM (P0.1)** – Input which specifies verify mode (output enable) or the program mode. OE/PGM = 1 output enabled (verify mode). OE/PGM = 0 program mode.
	8	9	I	**ASEL (P0.0)** – Input which indicates which bits of the EPROM address are applied to port 3. ASEL = 0 low address byte available on port 3. ASEL = 1 high address byte available on port 3 (only the three least significant bits are used).
	7	8	I/O	**SDA (P0.1)** – I^2C data.
	8	9	I/O	**SCL (P0.0)** – I^2C clock.
P1.0–P1.7	13–20	15–20, 23, 24	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 1 serves to output the addressed EPROM contents in the verify mode and accepts as inputs the value to program into the selected address during the program mode. Port 1 also serves the special function features of the 80C51 family as listed below:
	18	20	I	**$\overline{INT0}$ (P1.5):** External interrupt.
	19	23	I	**$\overline{INT1}$ (P1.6):** External interrupt.
	20	24	I	**T0 (P1.7):** Timer 0 external input.
P3.0–P3.7	5–1, 23–21	4–1, 6, 27–25	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also functions as the address input for the EPROM memory location to be programmed (or verified). The 11-bit address is multiplexed into this port as specified by P0.0/ASEL.
RST	9	11	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on RESET using only an external capacitor to V_{CC}. After the device is reset, a 10-bit serial sequence, sent LSB first, applied to RESET, places the device in the programming state allowing programming address, data and V_{PP} to be applied for programming or verification purposes. The RESET serial sequence must be synchronized with the X1 input.
X1	11	13	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits. X1 also serves as the clock to strobe in a serial bit stream into RESET to place the device in the programming state.
X2	10	12	O	**Crystal 2:** Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

X1 and X2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator.

To drive the device from an external clock source, X1 should be driven while X2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-up reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-up, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. the control bits for the reduced power modes are in the special function register PCON.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	Port 0	Port 1	Port 2
Idle	Data	Data	Data
Power-down	Data	Data	Data

DIFFERENCES BETWEEN THE 8XC751 AND THE 80C51

Program Memory

On the 8XC751, program memory is 2048 bytes long and is not externally expandable, so the 80C51 instructions MOVX, LJMP, and LCALL are not implemented. The only fixed locations in program memory are the addresses at which execution is taken up in response to reset and interrupts, which are as follows:

Event	Program Memory Address
Reset	000
External $\overline{INT0}$	003
Counter/timer 0	00B
External $\overline{INT1}$	013
Timer I	01B
I^2C serial	023

Counter/Timer Subsystem

The 8XC751 has one counter/timer called timer/counter 0. Its operation is similar to mode 2 operation on the 80C51, but is extended to 16 bits with 16 bits of autoload. The controls for this counter are centralized in a single register called TCON.

A watchdog timer, called Timer I, is for use with the I^2C subsystem. In I^2C applications, this timer is dedicated to time-generation and bus monitoring of the I^2C. In non-I^2C applications, it is available for use as a fixed time-base.

Interrupt Subsystem – Fixed Priority

The IP register and the 2-level interrupt system of the 80C51 are eliminated. Simultaneous interrupt conditions are resolved by a single-level, fixed priority as follows:

Highest priority:	Pin $\overline{INT0}$
	Counter/timer flag 0
	Pin $\overline{INT1}$
	Timer I
Lowest priority:	Serial I^2C

Serial Communications

The 8XC751 contains an I^2C serial communications port instead of the 80C51 UART. The I^2C serial port is a single bit hardware interface with all of the hardware necessary to support multimaster and slave operations. Also included are receiver digital filters and timer (timer I) for communication watch-dog purposes. The I^2C serial port is controlled through four special function registers; I^2C control, I^2C data, I^2C status, and I^2C configuration.

Special Function Register Addresses

Special function registers for the 8XC751 are identical to those of the 80C51, except for the changes listed below:

80C51 special function registers not present in the 8XC751 are TMOD (89), P2 (A0) and IP (B8). The 80C51 registers TH1, TL1, SCON, and SBUF are replaced with the 8XC751 registers RTH, RTL, I2CON, and I2DAT, respectively. Additional special function registers are I2CFG (D8) and I2STA (F8). See Table 2.

Table 2. I^2C Special Function Register Addresses

REGISTER ADDRESS			BIT ADDRESS							
NAME	SYMBOL	ADDRESS	MSB							LSB
I^2C control	I2CON	98	9F	9E	9D	9C	9B	9A	99	98
I^2C data	I2DAT	99	–	–	–	–	–	–	–	–
I^2C configuration	I2CFG	D8	DF	DE	DD	DC	DB	DA	D9	D8
I^2C status	I2STA	F8	FF	FE	FD	FC	FB	FA	F9	F8

ABSOLUTE MAXIMUM RATINGS[1, 2]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage from V_{CC} to V_{SS}	–0.5 to +6.5	V
Voltage from any pin to V_{SS} (except V_{PP})	–0.5 to V_{CC} + 0.5	V
Power dissipation	1.0	W
Voltage on V_{PP} pin to V_{SS}	0 to +13.0	V
Maximum I_{OL} per I/O pin	10	mA

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.

DC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10% for 87C751, V_{CC} = 5V ±20% for 83C751, V_{SS} = 0V[1]

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{IL}	Input low voltage, except SDA, SCL		–0.5	$0.2V_{DD}$–0.1	V
V_{IH}	Input high voltage, except X1, RST		$0.2V_{CC}$+0.9	V_{CC}+0.5	V
V_{IH1}	Input high voltage, X1, RST		$0.7V_{CC}$	V_{CC}+0.5	V
	SDA, SCL:				
V_{IL1}	Input low voltage		–0.5	$0.3V_{CC}$	V
V_{IH2}	Input high voltage		$0.7V_{CC}$	V_{CC}+0.5	V
V_{OL}	Output low voltage, ports 1 and 3	I_{OL} = 1.6mA[2]		0.45	V
V_{OL1}	Output low voltage, port 0.2	I_{OL} = 3.2mA[2]		0.45	V
V_{OH}	Output high voltage, ports 1 and 3	I_{OH} = –60µA	2.4		V
		I_{OH} = –25µA	$0.75V_{CC}$		V
		I_{OH} = –10µA	$0.9V_{CC}$		V
	Port 0.0 and 0.1 (I^2C) – Drivers				
V_{OL2}	Output low voltage	I_{OL} = 3mA		0.4	V
	Driver, receiver combined:	(over V_{CC} range)			
C	Capacitance			10	pF
I_{IL}	Logical 0 input current, ports 1 and 3	V_{IN} = 0.45V		–50	µA
I_{TL}	Logical 1 to 0 transition current, ports 1 and 3[3]	V_{IN} = 2V		–650	µA
I_{LI}	Input leakage current, port 0	0.45 < V_{IN} < V_{CC}		±10	µA
R_{RST}	Internal pull-down resistor		25	175	kΩ
C_{IO}	Pin capacitance	Test freq = 1MHz, T_{amb} = 25°C		10	pF
I_{PD}	Power-down current[4]	V_{CC} = 2 to V_{CC} max		50	µA

DC ELECTRICAL CHARACTERISTICS (Continued)

T_{amb} = 0°C to +70°C or −40°C to +85°C, V_{CC} = 5V ±10% for 87C751, V_{CC} = 5V ±20% for 83C751, V_{SS} = 0V[1]

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{PP}	V_{PP} program voltage (for 87C751 only)	V_{SS} = 0V V_{CC} = 5V±10% T_{amb} = 21°C to 27°C	12.5	13.0	V
I_{PP}	Program current (for 87C751 only)	V_{PP} = 13.0V		50	mA
I_{CC}	Supply current (see Figure 3)				

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.
2. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 10mA (NOTE: This is 85°C spec.)
   Maximum I_{OL} per 8-bit port: 26mA
   Maximum total I_{OL} for all outputs: 67mA
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
3. Pins of ports 1 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
4. Power-down I_{CC} is measured with all output pins disconnected; port 0 = V_{CC}; X2, X1 n.c.; RST = V_{SS}.
5. Active I_{CC} is measured with all output pins disconnected; X1 driven with t_{CLCH}, t_{CHCL} = 5ns, V_{IL} = V_{SS} + 0.5V, V_{IH} = V_{CC} − 0.5V; X2 n.c.; RST = port 0 = V_{CC}. I_{CC} will be slightly higher if a crystal oscillator is used.
6. Idle I_{CC} is measured with all output pins disconnected; X1 driven with t_{CLCH}, t_{CHCL} = 5ns, V_{IL} = V_{SS} + 0.5V, V_{IH} = V_{CC} − 0.5V; X2 n.c.; port 0 = V_{CC}; RST = V_{SS}.

AC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or −40°C to +85°C, V_{CC} = 5V ±10% for 87C751, V_{CC} = 5V ±20% for 83C751, V_{SS} = 0V[1,2]

SYMBOL	PARAMETER	12MHz CLOCK MIN	12MHz CLOCK MAX	VARIABLE CLOCK MIN	VARIABLE CLOCK MAX	UNIT
$1/t_{CLCL}$	Oscillator frequency:			3.5 3.5 0.5	12 16 12	MHz MHz MHz
External Clock (Figure 1)						
t_{CHCX}	High time	20		20		ns
t_{CLCX}	Low time	20		20		ns
t_{CLCH}	Rise time		20		20	ns
t_{CHCL}	Fall time		20		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.
2. Load capacitance for ports = 80pF.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

C – Clock
D – Input data
H – Logic level high
L – Logic level low
Q – Output data
T – Time
V – Valid
X – No longer a valid logic level
Z – Float

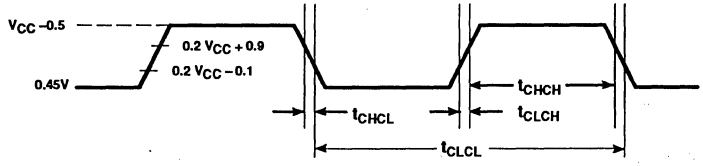


Figure 1. External Clock Drive

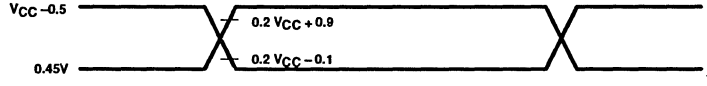


Figure 2. AC Testing Input/Output

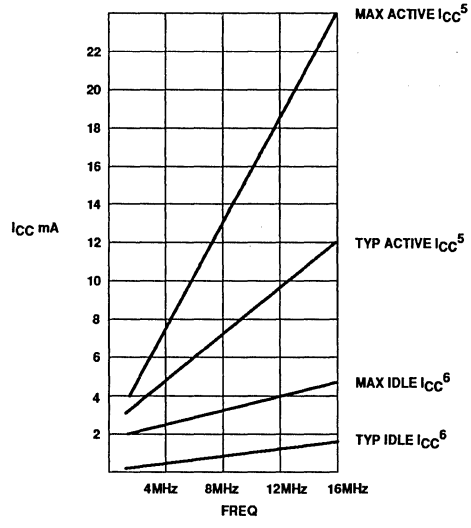


Figure 3. I_{CC} vs. FREQ
Maximum I_{CC} values taken at V_{CC} max and worst case temperature.
Typical I_{CC} values taken at V_{CC} = 5.0V and 25°C.
Notes 5 and 6 refer to DC Electrcial Characteristics.

PROGRAMMING CONSIDERATIONS

EPROM Characteristics

The 87C751 is programmed by using a modified Quick-Pulse Programming algorithm similar to that used for devices such as the 87C451 and 87C51. It differs from these devices in that a serial data stream is used to place the 87C751 in the programming mode.

Figure 4 shows a block diagram of the programming configuration for the 87C751. Port pin P0.2 is used as the programming voltage supply input (V_{PP} signal). Port pin P0.1 is used as the program (PGM/) signal. This pin is used for the 25 programming pulses.

Port 3 is used as the address input for the byte to be programmed and accepts both the high and low components of the eleven bit address. Multiplexing of these address components is performed using the ASEL input. The user should drive the ASEL input high and then drive port 3 with the high order bits of the address. ASEL should remain high for at least 13 clock cycles. ASEL may then be driven low which latches the high order bits of the address internally. the high address should remain on port 3 for at least two clock cycles after ASEL is driven low. Port 3 may then be driven with the low byte of the address. The low address will be internally stable 13 clock cycles later. The address will remain stable provided that the low byte placed on port 3 is held stable and ASEL is kept low. **Note:** ASEL needs to be pulsed high only to change the high byte of the address.

Port 1 is used as a bidirectional data bus during programming and verify operations. During programming mode, it accepts the byte to be programmed. During verify mode, it provides the contents of the EPROM location specified by the address which has been supplied to Port 3.

The XTAL1 pin is the oscillator input and receives the master system clock. This clock should be between 1.2 and 6MHz.

The RESET pin is used to accept the serial data stream that places the 87C751 into various programming modes. This pattern consists of a 10-bit code with the LSB sent first. Each bit is synchronized to the clock input, X1.

Programming Operation

Figures 5 and 6 show the timing diagrams for the program/verify cycle. RESET should initially be held high for at least two machine cycles. P0.1 (PGM/) and P0.2 (V_{PP}) will be at V_{OH} as a result of the RESET operation. At this point, these pins function as normal quasi-bidirectional I/O ports and the programming equipment may pull these lines low. However, prior to sending the 10-bit code on the RESET pin, the programming equipment should drive these pins high (V_{IH}). The RESET pin may now be used as the serial data input for the data stream which places the 87C751 in the programming mode. Data bits are sampled during the clock high time and thus should only change during the time that the clock is low. Following transmission of the last data bit, the RESET pin should be held low.

Next the address information for the location to be programmed is placed on port 3 and ASEL is used to perform the address multiplexing, as previously described. At this time, port 1 functions as an output.

A high voltage V_{PP} level is then applied to the V_{PP} input (P0.2). (This sets Port 1 as an input port). The data to be programmed into the EPROM array is then placed on Port 1. This is followed by a series of programming pulses applied to the PGM/ pin (P0.1). These pulses are created by driving P0.1 low and then high. This pulse is repeated until a total of 25 programming pulses have occurred. At the conclusion of the last pulse, the PGM/ signal should remain high.

The V_{PP} signal may now be driven to the V_{OH} level, placing the 87C751 in the verify mode. (Port 1 is now used as an output port). After four machine cycles (48 clock periods), the contents of the addressed location in the EPROM array will appear on Port 1.

The next programming cycle may now be initiated by placing the address information at the inputs of the multiplexed buffers, driving the V_{PP} pin to the V_{PP} voltage level, providing the byte to be programmed to Port1 and issuing the 26 programming pulses on the PGM/ pin, bringing V_{PP} back down to the V_C level and verifying the byte.

Programming Modes

The 87C751 has four programming features incorporated within its EPROM array. These include the USER EPROM for storage of the application's code, a 16-byte encryption key array and two security bits. Programming and verification of these four elements are selected by a combination of the serial data stream applied to the RESET pin and the voltage levels applied to port pins P0.1 and P0.2. The various combinations are shown in Table 3.

Encryption Key Table

The 87C751 includes a 16-byte EPROM array that is programmable by the end user. The contents of this array can then be used to encrypt the program memory contents during a program memory verify operation. When a program memory verify operation is performed, the contents of the program memory location is XNOR'ed with one of the bytes in the 16-byte encryption table. The resulting data pattern is then provided to port 1 as the verify data. The encryption mechanism can be disable, in essence, by leaving the bytes in the encryption table in their erased state (FFH) since the XNOR product of a bit with a logical one will result in the original bit. The encryption bytes are mapped with the code memory in 16-byte groups. the first byte in code memory will be encrypted with the first byte in the encryption table; the second byte in code memory will be encrypted with the second byte in the encryption table and so forth up to and including the 16the byte. The encryption repeats in 16-byte groups; the 17th byte in the code memory will be encrypted with the first byte in the encryption table, and so forth.

Security Bits

Two security bits, security bit 1 and security bit 2, are provided to limit access to the USER EPROM and encryption key arrays. Security bit 1 is the program inhibit bit, and once programmed performs the following functions:

1. Additional programming of the USER EPROM is inhibited.
2. Additional programming of the encryption key is inhibited.
3. Verification of the encryption key is inhibited.
4. Verification of the USER EPROM and the security bit levels may still be performed.

(If the encryption key array is being used, this security bit should be programmed by the user to prevent unauthorized parties from reprogramming the encryption key to all logical zero bits. Such programming would provide data during a verify cycle that is the logical complement of the USER EPROM contents).

Security bit 2, the verify inhibit bit, prevents verification of both the USER EPROM array and the encryption key arrays. The security bit levels may still be verified.

Programming and Verifying Security Bits

Security bits are programmed employing the same techniques used to program the USER EPROM and KEY arrays using serial data streams and logic levels on port pins indicated in Table 3. When programming either security bit, it is not necessary to provide address or data information to the 87C751 on ports 1 and 3.

Verification occurs in a similar manner using the RESET serial stream shown in Table 3. Port 3 is not required to be driven and the results of the verify operation will appear on ports 1.6 and 1.7.

Ports 1.7 contains the security bit 1 data and is a logical one if programmed and a logical zero if erased. Likewise, P1.6 contains the security bit 2 data and is a logical one if programmed and a logical zero if erased.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Flourless part number 2345–5 or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-s/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000µW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 3. Implementing Program/Verify Modes

OPERATION	SERIAL CODE	P0.1 (PGM/)	P0.2 (V_{PP})
Program user EPROM	296H	–*	V_{PP}
Verify user EPROM	296H	V_{IH}	V_{IH}
Program key EPROM	292H	–*	V_{PP}
Verify key EPROM	292H	V_{IH}	V_{IH}
Program security bit 1	29AH	–*	V_{PP}
Program security bit 2	298H	–*	V_{PP}
Verify security bits	29AH	V_{IH}	V_{IH}

NOTE:
* Pulsed from V_{IH} to V_{IL} and returned to V_{IH}.

EPROM PROGRAMMING AND VERIFICATION

T_{amb} = 21°C to +27°C, V_{CC} = 5V ±10%, V_{SS} = 0V

SYMBOL	PARAMETER	MIN	MAX	UNIT
$1/t_{CLCL}$	Oscillator/clock frequency	1.2	6	MHz
t_{AVGL}*	Address setup to P0.1 (PROG–) low	10µs + 24t_{CLCL}		
t_{GHAX}	Address hold after P0.1 (PROG–) high	48t_{CLCL}		
t_{DVGL}	Data setup to P0.1 (PROG–) low	38t_{CLCL}		
t_{DVGL}	Data setup to P0.1 (PROG–) low	38t_{CLCL}		
t_{GHDX}	Data hold after P0.1 (PROG–) high	36t_{CLCL}		
t_{SHGL}	V_{PP} setup to P0.1 (PROG–) low	10		µs
t_{GHSL}	V_{PP} hold after P0.1 (PROG–)	10		µs
t_{GLGH}	P0.1 (PROG–) width	90	110	µs
t_{AVQV}**	V_{PP} low (V_{CC}) to data valid		48t_{CLCL}	
t_{GHGL}	P0.1 (PROG–) high to P0.1 (PROG–) low	10		µs
t_{SYNL}	P0.0 (sync pulse) low	4t_{CLCL}		
t_{SYNH}	P0.0 (sync pulse) high	8t_{CLCL}		
t_{MASEL}	ASEL high time	13t_{CLCL}		
t_{MAHLD}	Address hold time	2t_{CLCL}		
t_{HASET}	Address setup to ASEL	13t_{CLCL}		
t_{ADSTA}	Low address to valid data		48t_{CLCL}	

NOTES:
* Address should be valid at least 24t_{CLCL} before the rising edge of P0.2 (V_{PP}).
** For a pure verify mode, i.e., no program mode in between, t_{AVQV} is 14t_{CLCL} maximum.

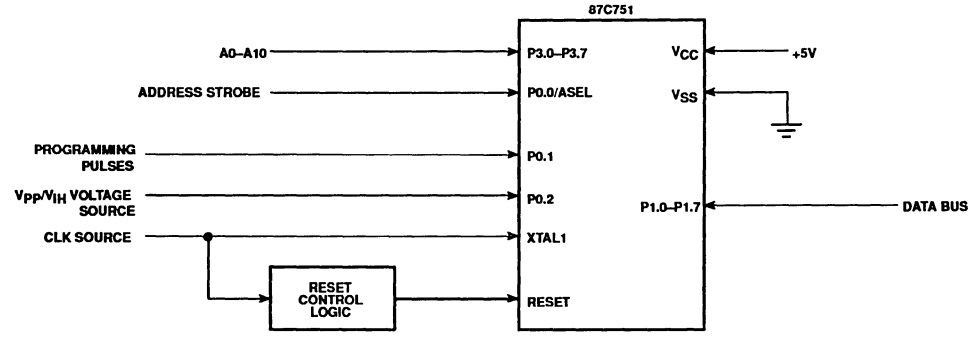


Figure 4. Programming Configuration

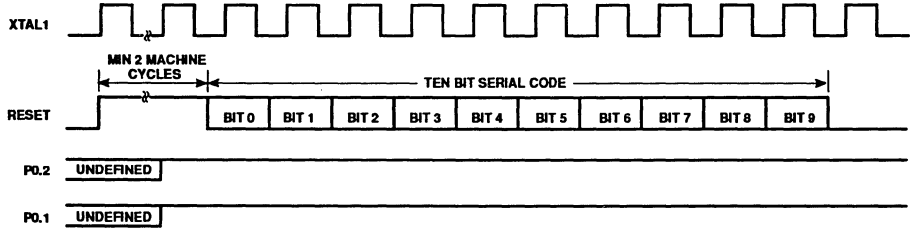


Figure 5. Entry into Program/Verify Modes

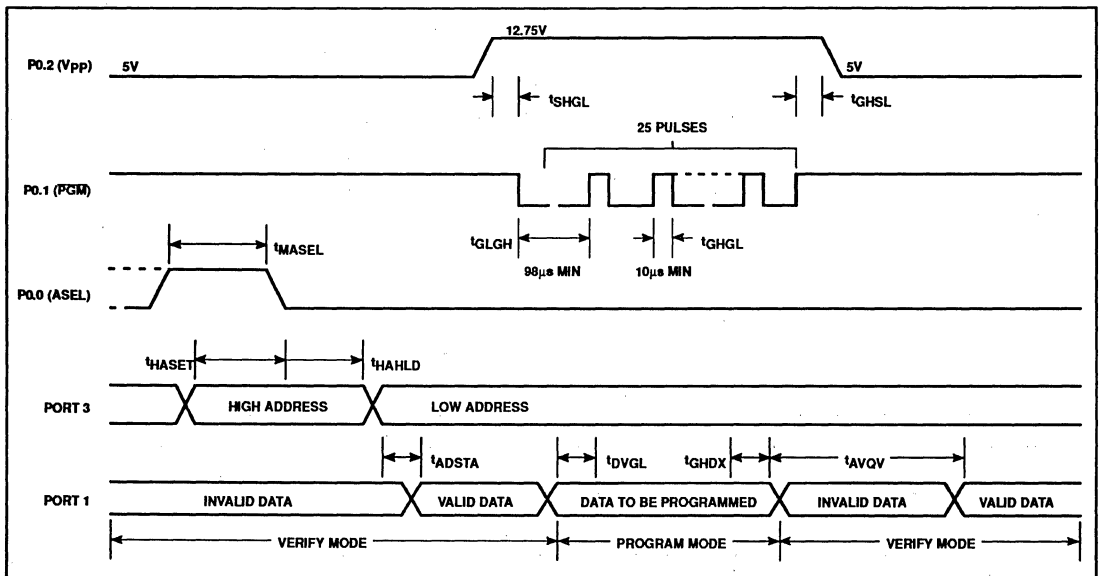


Figure 6. Program/Verify Cycle

Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specifications defined by Philips.

8XC752 OVERVIEW

The Signetics 83C752/87C752 is a single-chip control oriented microcontroller fabricated with Signetics high-density CMOS technology minimizing CMOS latch-up sensitivity. Being a member of the 80C51 family, the 83C752 has a powerful instruction set, and has the same basic architecture as the 80C51. The 83C752 is essentially the popular industry-standard 83C751 with the inclusion of a five-channel multiplexed 8-bit ADC and a PWM output.

The 83C752 contains a 2k × 8 masked ROM, 64 bytes of RAM, 21 I/O lines, a 16-bit auto-reload timer/counter, a fixed-rate timer, a seven-source fixed-priority interrupt structure, a bidirectional Inter-Integrated Circuit (I^2C) serial bus interface, and an on-chip oscillator. This device also includes a five-channel multiplexed 8-bit ADC and a PWM output.

The on-board I^2C bus interface allows the 83C752 to operate as a master or slave device on the I^2C small area network. This capability facilitates I/O and RAM expansion, access to EEPROM, processor-to-processor communications, and efficient interface to a wide variety of dedicated I^2C peripherals.

The EPROM version of this device, the 87C752, is also available in both quartz-lid erasable and plastic one-time programmable (OTP) packages. Once the array has been programmed, it is functionally equivalent to the masked ROM 83C752. Thus, unless explicitly stated otherwise, all references made to the 83C752 apply equally to the 87C752.

The 83C752 supports two power reduction modes of operation referred to as the idle mode and the power-down mode.

Differences from the 80C51

Instruction Set

The instruction set of the 83C752 is identical to the 80C51 except that:

MOVX, LCALL, and LJMP are not implemented.

If these instructions are executed, the appropriate number of instruction cycles will take place along with external fetches; however, no operation will take place. The LJMP may not respond to all program address bits.

Memory Organization

The 83C752 manipulates operands in three memory address spaces. The first is the program memory space which contains program instructions as well as constants such as look-up tables. The program memory space contains 2k bytes in the 83C752.

The second memory space is the data memory array which has a logical address space of 128 bytes. However, only the first 64 (0 to 3FH) are implemented in the 83C752.

The third memory space is the special function register array having a 128-byte address space (80H to FFH). Only selected locations in this memory space are used (see Table 1). Note that the architecture of these memory spaces (internal program memory, internal data memory, and special function registers) is identical to the 80C51, and the 83C752 varies only in the amount of memory physically implemented.

The 83C752 does not directly address any external data or program memory spaces. For this reason, the MOVX instructions in the 80C51 instruction set are not implemented in the 83C752, nor are the alternate I/O pin functions $\overline{RD}$ and $\overline{WR}$.

I/O Ports

The I/O pins provided by the 83C752 consist of port 0, port 1, and port 3.

Port 0

Port 0 is a 5-bit bidirectional I/O port and includes alternate functions on some pins of this port. Pins P0.3 and P0.4 are provided with internal pullups while the remaining pins (P0.0, P0.1, and P0.2) have open drain output structures. The alternate functions for port 0 are:

P0.0 SCL – the I^2C bus clock
P0.1 SDA – the I^2C bus data
P0.4 PWM – the PWM output

If the alternate functions, I^2C and PWM, are not being used, then these pins may be used as I/O ports.

Port 1

Port 1 is an 8-bit bidirectional I/O port whose structure is identical to the 80C51, but also includes alternate input functions on all pins. The alternate pin functions for port 1 are:

P1.0-P1.4 - ADC0-ADC4 - A/D converter analog inputs
P1.5 $\overline{INT0}$ - external interrupt 0 input
P1.6 $\overline{INT1}$ - external interrupt 1 input
P1.7 - T0 - timer 0 external input

If the alternate functions $\overline{INT0}$, $\overline{INT1}$, or T0 are not being used, these pins may be used as standard I/O ports, provided that the A/D is disabled. It is necessary to connect AV_{CC} and AV_{SS} to V_{CC} and V_{SS}, respectively, in order to use these pins as standard I/O pins. When the A/D converter is enabled, the analog channel connected to the A/D may not be used as a digital input; however, the remaining analog inputs may be used as digital inputs. They may not be used as digital outputs. While the A/D is enabled, the analog inputs are floating.

Port 3

Port 3 is an 8-bit bidirectional I/O port whose structure is identical to the 80C51. Note that the alternate functions associated with port 3 of the 80C51 have been moved to port 1 of the 83C752 (as applicable). See Figure 1 for port bit configurations.

PWM Outputs

The single PWM output is an alternate function assigned to P0.4 and can be used to output pulses of programmable length and interval. The repetition frequency is defined by an 8-bit prescaler which generates the clock for the counter. This prescaler is contained in the PWMP register.

The 8-bit counter counts from 0 to 254 inclusive. The value of the 8-bit counter is compared to the contents of the compare register, PWM. When the content's value matches that of the PWCM register, the PWCM output is set high. When the counter reaches zero, the PWM output is set low. The pulse width ratio (duty cycle) is defined by the contents of the compare register and is in the range of 0 to 1, programmed in increments of 1/255.

Table 1. 8XC752 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
ADAT#	A/D result	84H									00H
ADCON*#	A/D control	A0H	–	–	ENADC	ADCI	ADCS	AADR2	AADR1	AADR0	C0H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data pointer (2 bytes)										
DPL	Data pointer low	82H									00H
DPH	Data pointer high	83H									00H
			DF	DE	DD	DC	DB	DA	D9	D8	
I²CFG*#	I²C configuration	D8H/RD	SLAVEN	MASTRQ	0	TIRUN	–	–	CT1	CT0	0000xx00B
		WR	SLAVEN	MASTRQ	CLRTI	TIRUN	–	–	CT1	CT0	
			9F	9E	9D	9C	9B	9A	99	98	
I²CON*#	I²C control	98H/RD	RDAT	ATN	DRDY	ARL	STR	STP	MASTER	–	81H
		WR	CXA	IDLE	CDR	CARL	CSTR	CSTP	XSTR	XSTP	
I²DAT*#	I²C data	99H/RD	RDAT	0	0	0	0	0	0	0	80H
		WR	XDAT	X	X	X	X	X	X	X	
			FF	FE	FD	FC	FB	FA	F9	F8	
I²STA*#	I²C status	F8H	–	IDLE	XDATA	XACTV	MAKSTR	MAKSTP	XSTR	XSTP	x0100000B
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*#	Interrupt enable	A8H	EA	EAD	ETI	ES	EPWM	EX1	ET0	EX0	00H
			–	–	–	84	83	82	81	80	xxx11111B
P0*#	Port 0	80H	–	–	–	PWM0	–	–	SDA	SCL	
			97	96	95	94	93	92	91	90	FFH
P1*#	Port 1	90H	T0	$\overline{INT1}$	INT0	ADC4	ADC3	ADC2	ADC1	ADC0	
P3*	Port 3	B0H	B7	B6	B5	B4	B3	B2	B1	B0	FFH
PCON#	Power control	87H	–	–	–	–	–	–	PD	IDL	xxxx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
PWCM#	PWM compare	8EH									xxxxxxxxB
PWENA#	PWM enable	FEH	–	–	–	–	–	–	–	PWE	FEH
PWMP#	PWM prescaler	8FH									00H
RTL#	Timer low reload	8BH									00H
RTH#	Timer high reload	8DH									00H
SP	Stack pointer	81H									07H
TL#	Timer low	8AH									00H
TH#	Timer high	8CH									00H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*#	Timer control	88H	GATE	C/T	TF	TR	IE0	IT0	IE1	IT1	00H

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.

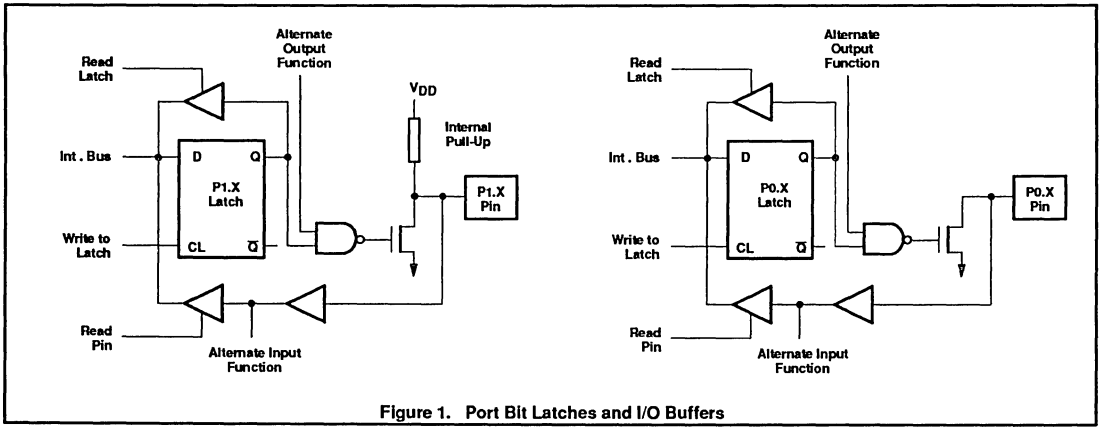


Figure 1. Port Bit Latches and I/O Buffers

The PWM output can be set continuously high by loading the compare register with 00H and continuously low by loading the compare register with FFH. The PWM output is enabled by setting the PWE bit in the PWM enable register, PWENA. When enabled, the output is driven with a fully active strong pullup. When disabled, the pin behaves as a normal bidirectional I/O pin. When disabled, the counter remains active. The PWM function is disabled by a reset condition. The PWM output is high during power-down and idle modes and the counter is disabled.

The repetition frequency is given by:

$$f_{PWM} = \frac{f_{OSC}}{510 \times (1 + PWMP)}$$

An oscillator frequency of 12MHz results in a repetition range of 92Hz to 23.5kHz.

The low/high ratio of the PWM output is PWM/(255 – PWM) for PWM values except 255. A PWM value of 255 results in a low PWM output.

If enabled, a PWM interrupt will occur when the PWM counter overflows.

In order for the PWM output to be used as a standard I/O pin, the PWM function needs to be disabled. The PWM counter can still be used as an internal timer by enabling the PWM interrupt.

A/D Converter

The 83C752 contains a five-channel multiplexed 8-bit A/D converter. The conversion requires 40 machine cycles (40µs at 12MHz oscillator frequency).

The A/D converter is controlled by the A/D control register, ADCON. Input channels are selected by the analog multiplexer by bits ADCON.0 through ADCON.2. The ADCON register is not bit addressable.

ADCON Register

MSB							LSB
X	X	ENADC	ADCI	ADCS	AADR2	AADR1	AADR0

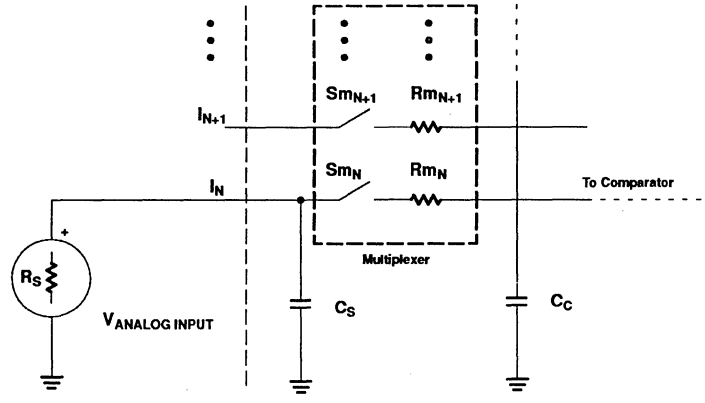
ADCI	ADCS	Operation
0	0	ADC not busy, a conversion can be started.
0	1	ADC busy, start of a new conversion is blocked.
1	0	Conversion completed, start of a new conversion is blocked.
1	1	Not possible.

INPUT CHANNEL SELECTION			
ADDR2	ADDR1	ADDR0	INPUT PIN
0	0	0	P1.0
0	0	1	P1.1
0	1	0	P1.2
0	1	1	P1.3
1	0	0	P1.4

Position	Symbol	Function
ADCON.5	ENADC	Enable A/D function when ENADC = 1. Reset forces ENADC = 0.
ADCON.4	ADCI	ADC interrupt flag. This flag is set when an ADC conversion is complete. If IE.6 = 1, an interrupt is requested when ADCI = 1. The ADCI flag is cleared when conversion data is read. This flag is read only.
ADCON.3	ADCS	ADC start. Setting this bit starts an A/D conversion. Once set, ADCS remains high throughout the conversion cycle. On completion of the conversion, it is reset just before the ADCI interrupt flag is cleared. ADCS cannot be reset by software. ADCS should not be used to monitor the A/D converter status. ADCI should be used for this purpose.
ADCON.2	AADR2	Analog input select.
ADCON.1	AADR1	Analog input select.
ADCON.0	AADR0	Analog input select. This binary coded address selects one of the five analog input port pins of P1 to be input to the converter. It can only be changed when ADCI and ADCS are both low. AADR2 is the most significant bit.

The completion of the 8-bit ADC conversion is flagged by ADCI in the ADCON register, and the result is stored in the special function register ADAT.

An ADC conversion in progress is unaffected by an ADC start. The result of a completed conversion remains unaffected provided ADCI remains at a logic 1. While ADCS is a logic 1 or ADCI is a logic 1, a new ADC START will be blocked and consequently lost. An ADC conversion in progress is aborted when the idle or power-down mode is entered. The result of a completed conversion (ADCI = logic 1) remains unaffected when entering the idle mode. See Figure 2 for an A/D input equivalent circuit.



Rm = 0.5 - 3 kohms
CS + CC = 15pF maximum
RS = Recommended < 9.6 kohms for 1 LSB @ 12MHz

NOTE:
Because the analog to digital converter has a sampled-data comparator, the input looks capacitive to a source. When a conversion is initiated, switch Sm closes for 8tcy (8µs @ 12MHz crystal frequency) during which time capacitance Cs + Cc is charged. It should be noted that the sampling causes the analog input to present a varying load to an analog source.

Figure 2. A/D Input: Equivalent Circuit

The analog input pins ADC0-ADC4 may be used as digital inputs and outputs when the A/D converter is disabled by a 0 in the ENADC bit in ADCON. When the A/D is enabled, the analog input channel that is selected by the ADDR2-ADDR0 bits in ADCON cannot be used as a digital input. Reading the selected A/D channel as a digital input will always return a 1. The unselected A/D inputs may always be used as digital inputs. Unselected analog inputs will be floating and may not be used as digital outputs.

Counter/Timer

The 8XC752 counter/timer is designated Timer 0 and is separate from Timer I of the I²C serial port and from the PWM. Its operation is similar to mode 2 of the 80C51 counter/timer, extended to 16 bits. When Timer 0 is used in the external counter mode, the T0 input (P1.7) is sampled every S4P1. The counter/timer function is controlled using the timer control register (TCON).

TCON Register

MSB LSB

GATE	C/T	TF	TR	IE0	IT0	IE1	IT1

Position	Symbol	Function
TCON.7	GATE	1 – Timer 0 is enabled only when INT0 pin is high and TR is 1. 0 – Timer 0 is enabled only when TR is 1.
TCON.6	C/T	1 – Counter operation from T0 pin. 0 – Timer operation from internal clock.
TCON.5	TF	1 – Set on overflow of T0. 0 – Cleared when processor vectors to interrupt routine and by reset.
TCON.4	TR	1 – Enable timer 0 0 – Disable timer 0
TCON.3	IE0	1 – Edge detected on INT0
TCON.2	IT0	1 – INT0 is edge triggered. 0 – INT0 is level sensitive.
TCON.1	IE1	1 – Edge detected on INT1
TCON.0	IT1	1 – INT1 is edge triggered. 0 – INT1 is level sensitive.

These flags are functionally identical to the corresponding 80C51 flags except that there is only one of the 80C51 style timers, and the flags are combined into one register.

Note that the positions of the IE0/IT0 and IE1/IT1 bits are transposed from the positions used in the standard 80C51 TCON register.

A communications watchdog timer, Timer I, is described in the I²C section. In I²C applications, this timer is dedicated to time generation and bus monitoring for the I²C. In non-I²C applications, it is available for use as a fixed time base.

The 16-bit timer/counter's operation is similar to mode 2 operation on the 80C51, but is extended to 16 bits. The timer/counter is clocked by either 1/12 the oscillator frequency or by transitions on the T0 pin. The C/T pin in special function register TCON selects between these two modes. When the TCON TR bit is set, the timer/counter is enabled. Register pair TH and TL are incremented by the clock source. When the register pair overflows, the register pair is reloaded with the values in registers RTH and RTL. The value in the reload registers is left unchanged. The TF bit in special function register TCON is set on counter overflow and, if the interrupt is enabled, will generate an interrupt (see Figure 16).

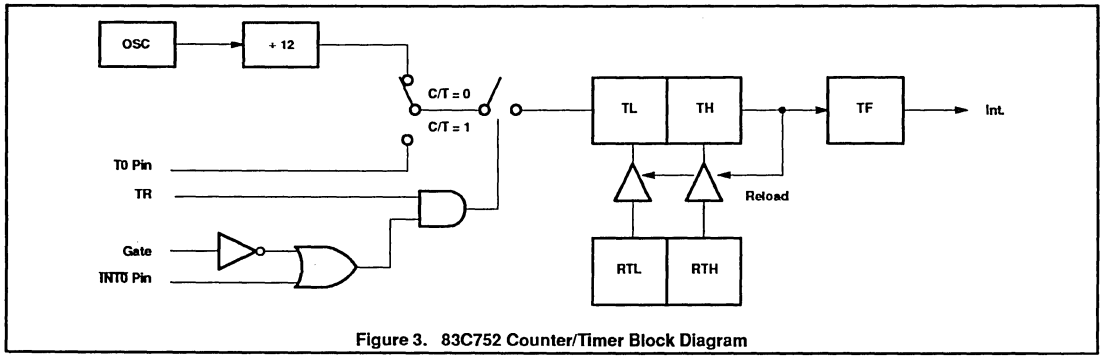


Figure 3. 83C752 Counter/Timer Block Diagram

I²C Serial I/O

The I²C bus uses two wires (SDA and SCL) to transfer information between devices connected to the bus. The main technical features of the bus are:

- Bidirectional data transfer between masters and slaves
- Serial addressing of slaves
- Acknowledgment after each transferred byte
- Multimaster bus
- Arbitration between simultaneously transmitting master without corruption of serial data on bus

A large family of I²C compatible ICs is available. See the I²C section for more details on the bus and available ICs.

The 83C752 I²C subsystem includes hardware to simplify the software required to drive the I²C bus. This circuitry is the same as that on the 83C751. (See the 83C751 section for a detailed discussion of this subsystem).

Interrupts

The interrupt structure is a seven-source, one-level interrupt system similar to the 8XC751. The interrupt sources are listed below in their order of polling sequence priority (highest to lowest):

Priority	Source	Function
Highest	INT0	External interrupt 0
	TF0	Timer flag 0
	INT1	External interrupt 1
	PWM	PWM counter overflow
	TI	I²C timer overflow
	SIO	Serial port interrupt
Lowest	ADC	A/D conversion complete

The vector addresses are as follows:

Source	Vector Address
INT0	0003H
TF0	000BH
INT1	0013H
TIMER I	001BH
SIO	0023H
ADC	002BH
PWM	0033H

Interrupt Control Registers

The 80C51 interrupt enable register is modified to take into account the different interrupt sources of the 8XC752.

Interrupt Enable Register

MSB							LSB
EA	EAD	ETI	ES	EPWM	EX1	ET0	EX0

Position	Symbol	Function
IE.7	EA	Global interrupt disable when EA = 0
IE.6	EAD	A/D conversion complete
IE.5	ETI	Timer I
IE.4	ES	I²C serial port
IE.3	EPWM	PWM counter overflow
IE.2	EX1	External interrupt 1
IE.1	ET0	Timer 0 overflow
IE.0	EX0	External interrupt 0

Power-Down and Idle Modes

The 8XC752 includes the 80C51 power-down and idle mode features. The functions that continue to run while in the idle mode are Timer 0, the I²C interface including Timer I, and the interrupts. Upon powering-up the circuit, or exiting from idle mode, sufficient time must be allowed for stabilization of the internal analog reference voltages before an A/D conversion is started.

Special Function Registers

The special function registers (directly addressable only) contain all of the 8XC751 registers except the program counter and the four register banks. Most of the 21 special function registers are used to control the on-chip peripheral hardware. Other registers include arithmetic registers (ACC, B, PSW), stack pointer (SP) and data pointer registers (DPH, DPL). Nine of the SFRs are bit addressable.

Data Pointer

The data pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). In the 80C51 this register allows the access of external data memory using the MOVX instruction. Since the 83C752 does not support MOVX or external memory accesses, this register is generally used as a 16-bit offset pointer of the accumulator in a MOVC instruction. DPTR may also be manipulated as two independent 8-bit registers.

CMOS single-chip 8-bit microcontroller — 83C752/87C752

CMOS single-chip 8-bit microcontroller with A/D, PWM

DESCRIPTION

The Philips 83C752/87C752 offers many of the advantages of the 80C51 architecture in a small package and at low cost.

The 8XC752 Microcontroller is fabricated with Philips high-density CMOS technology. Philips epitaxial substrate minimizes CMOS latch-up sensitivity.

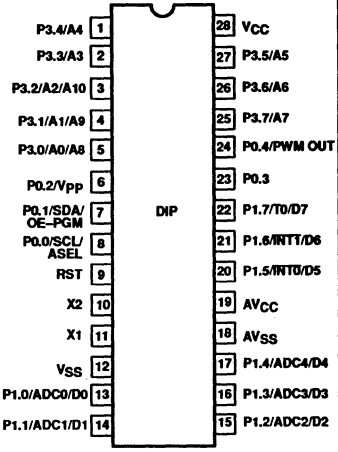
The 8XC752 contains a 2k × 8 ROM (83C752) EPROM (87C752), a 64 × 8 RAM, 21 I/O lines, a 16-bit auto-reload counter/timer, a fixed-priority level interrupt structure, a bidirectional inter-integrated circuit (I^2C) serial bus interface, an on-chip oscillator, a five channel multiplexed 8-bit A/D converter, and an 8-bit PWM output.

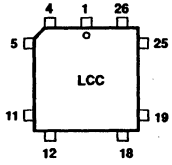
The onboard inter-integrated circuit (I^2C) bus interface allows the 8XC752 to operate as a master or slave device on the I^2C small area network. This capability facilitates I/O and RAM expansion, access to EEPROM, processor-to-processor communication, and efficient interface to a wide variety of dedicated I^2C peripherals.

FEATURES

- Available in erasable quartz lid or One-Time Programmable plastic packages
- 80C51 based architecture
- Inter-integrated Circuit (I^2C) serial bus interface
- Small package sizes
  - 28-pin DIP
  - 28-pin PLCC
- Wide oscillator frequency range
- Low power consumption:
  - Normal operation: less than 11mA @ 5V, 12MHz
  - Idle mode
  - Power-down mode
- 2k × 8 ROM (83C752) EPROM (87C752)
- 64 × 8 RAM
- 16-bit auto reloadable counter/timer
- 5-channel 8-bit A/D converter
- 8-bit PWM output/timer
- Fixed-rate timer
- Boolean processor
- CMOS and TTL compatible
- Well suited for logic replacement, consumer and industrial applications

PIN CONFIGURATION



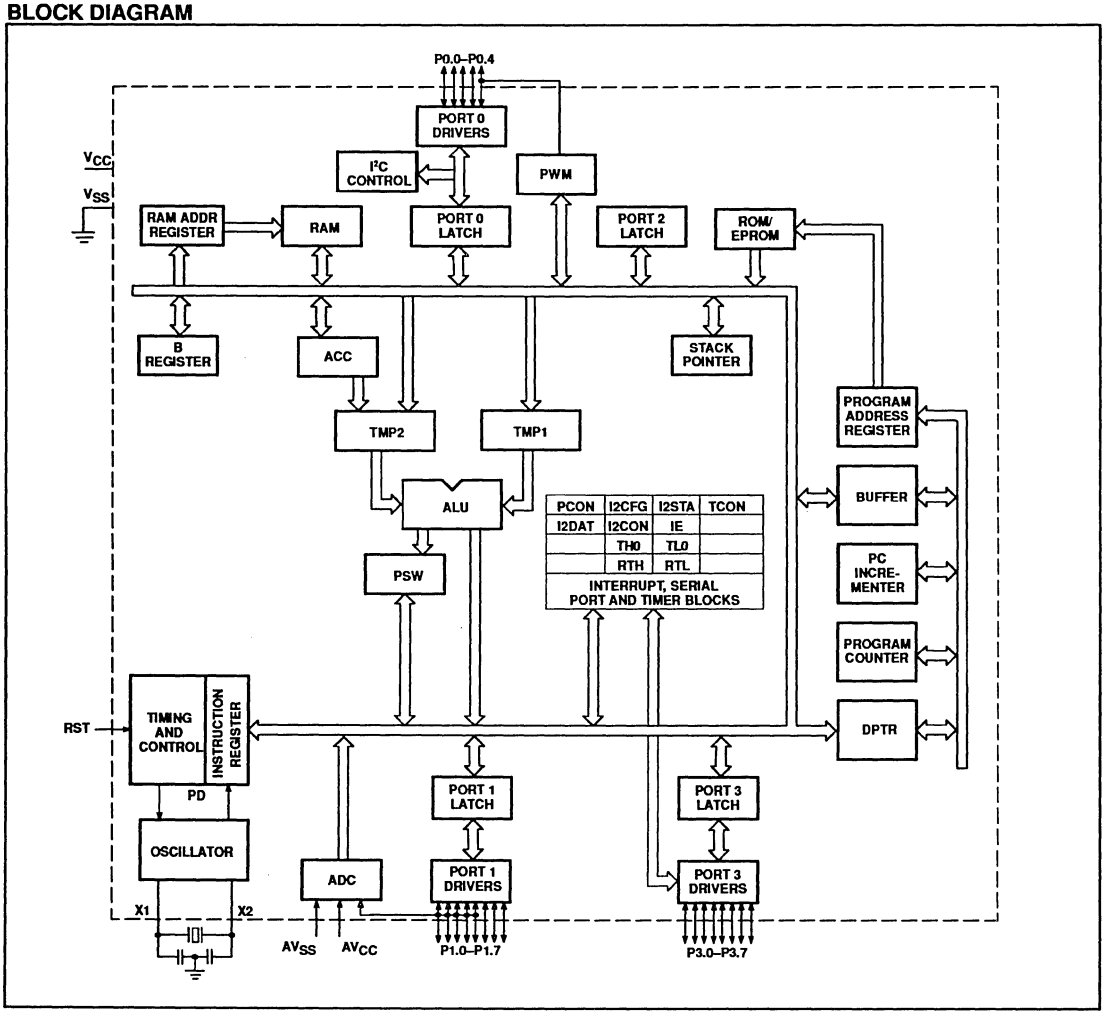


Pin	Function	Pin	Function
1	P3.4/A4	15	P1.2/ADC2/D2
2	P3.3/A3	16	P1.3/ADC3/D3
3	P3.2/A2/A10	17	P1.4/ADC4/D4
4	P3.1/A1/A9	18	AV_{SS}
5	P3.0/A0/A8	19	AV_{CC}
6	P0.2/V_{PP}	20	P1.5/INT0/D5
7	P0.1/SDA/OE–PGM	21	P1.6/INT1/D6
8	P0.0/SCL/ASEL	22	P1.7/T0/D7
9	RST	23	P0.3
10	X2	24	P0.4/PWM OUT
11	X1	25	P3.7/A7
12	V_{SS}	26	P3.6/A6
13	P1.0/ADC0/D0	27	P3.5/A5
14	P1.1/ADC1/D1	28	V_{CC}

PART NUMBER SELECTION

ROM	EPROM	SPEED	TEMPERATURE AND PACKAGE
	S87C752–1F28	3.5 to 12MHz	0 to +70°C, ceramic DIP
	S87C752–2F28	3.5 to 12MHz	–40 to +85°C, ceramic DIP
	S87C752–4F28	3.5 to 16MHz	0 to +70°C, ceramic DIP
	S87C752–5F28	3.5 to 16MHz	–40 to +85°C, ceramic DIP
S83C752–1N28	S87C752–1N28	3.5 to 12MHz	0 to +70°C, plastic DIP
S83C752–2N28	S87C752–2N28	3.5 to 12MHz	–40 to +85°C, plastic DIP
S83C752–4N28	S87C752–4N28	3.5 to 16MHz	0 to +70°C, plastic DIP
S83C752–5N28	S87C752–5N28	3.5 to 16MHz	–40 to +85°C, plastic DIP
S83C752–1A28	S87C752–1A28	3.5 to 12MHz	0 to +70°C, plastic LCC
S83C752–2A28	S87C752–2A28	3.5 to 12MHz	–40 to +85°C, plastic LCC
S83C752–4A28	S87C752–4A28	3.5 to 16MHz	0 to +70°C, plastic LCC
S83C752–5A28	S87C752–5A28	3.5 to 16MHz	–40 to +85°C, plastic LCC
S83C752–6A28	S87C752–6A28	3.5 to 12MHz	–55 to +125°C, plastic LCC
S83C752–6F28	S87C752–6F28	3.5 to 12MHz	–55 to +125°C, ceramic DIP
S83C752–6N28	S87C752–6N28	3.5 to 12MHz	–55 to +125°C, plastic DIP

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{SS}	12	I	**Circuit Ground Potential.**
V_{CC}	28	I	**Supply voltage during normal, idle, and power-down operation.**
P0.0–P0.4	8–6 23, 24	I/O	**Port 0:** Port 0 is a 5-bit bidirectional port. Port 0.0–P0.2 are open drain. Port 0.0–P0.2 pins that have 1s written to them float, and in that state can be used as high-impedance inputs. P0.3–P0.4 are bidirectional I/O port pins with internal pull-ups. Port 0 also serves as the serial I^2C interface. When this feature is activated by software, SCL and SDA are driven low in accordance with the I^2C protocol. These pins are driven low if the port register bit is written with a 0 or if the I^2C subsystem presents a 0. The state of the pin can always be read from the port register by the program. Port 0.3 and 0.4 have internal pull-ups that function identically to port 3. Pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. To comply with the I^2C specification, P0.0 and P0.1 are open drain bidirectional I/O pins with the electrical characteristics listed in the tables that follow. While these differ from "standard TTL" characteristics, they are close enough for the pins to still be used as general-purpose I/O in non-I^2C applications.
	6	I	**V_{PP} (P0.2) –** Programming voltage input.
	7	I	**OE/PGM (P0.1) –** Input which specifies verify mode (output enable) or the program mode. OE/PGM = 1 output enabled (verify mode). OE/PGM = 0 program mode.
	8	I	**ASEL (P0.0) –** Input which indicates which bits of the EPROM address are applied to port 3. ASEL = 0 low address byte available on port 3. ASEL = 1 high address byte available on port 3 (only the three least significant bits are used).
P1.0–P1.7	13–17, 20–22	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. P0.3–P0.4 pins are bidirectional I/O port pins with internal pull-ups. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 1 also serves the special function features of the SC80C51 family as listed below:
	20	I	**$\overline{INT0}$ (P1.5):** External interrupt.
	21	I	**$\overline{INT1}$ (P1.6):** External interrupt.
	22	I	**T0 (P1.7):** Timer 0 external input.
	13–17	I	**ADC0 (P1.0)–ADC4 (P1.4):** Port 1 also functions as the inputs to the five channel multiplexed A/D converter. These pins can be used as outputs only if the A/D function has been disabled. These pins can be used as inputs while the A/D converter is enabled. Port 1 serves to output the addressed EPROM contents in the verify mode and accepts as inputs the value to program into the selected address during the program mode.
P3.0–P3.7	5–1, 27–25	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also functions as the address input for the EPROM memory location to be programmed (or verified). The 11-bit address is multiplexed into this port as specified by P0.0/ASEL.
RST	9	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running resets the device. An internal diffused resistor to V_{SS} permits a power-on RESET using only an external capacitor to V_{CC}. After the device is reset, a 10-bit serial sequence, sent LSB first, applied to RESET, places the device in the programming state allowing programming address, data and V_{PP} to be applied for programming or verification purposes. The RESET serial sequence must be synchronized with the X1 input.
X1	11	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits. X1 also serves as the clock to strobe in a serial bit stream into RESET to place the device in the programming state.
X2	10	O	**Crystal 2:** Output from the inverting oscillator amplifier.
AV_{CC}	19	I	**Analog supply voltage and reference input.**
AV_{CC}	18	I	**Analog supply and reference ground.**

OSCILLATOR CHARACTERISTICS

X1 and X2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator.

To drive the device from an external clock source, X1 should be driven while X2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. The control bits for the reduced power modes are in the special function register PCON.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	Port 0*	Port 1	Port 2
Idle	Data	Data	Data
Power-down	Data	Data	Data

* Except for PWM output (P0.4).

DIFFERENCES BETWEEN THE 8XC752 AND THE 80C51

Program Memory

On the 8XC752, program memory is 2048 bytes long and is not externally expandable, so the 80C51 instructions MOVX, LJMP, and LCALL are not implemented. The only fixed locations in program memory are the addresses at which execution is taken up in response to reset and interrupts, which are as follows:

Event	Program Memory Address
Reset	000
External $\overline{INT0}$	003
Counter/timer 0	00B
External $\overline{INT1}$	013
Timer I	01B
I^2C serial	023
ADC	02B
PWM	033

Counter/Timer Subsystem

The 8XC752 has one counter/timer called timer/counter 0. Its operation is similar to mode 2 operation on the 80C51, but is extended to 16 bits with 16 bits of autoload. The controls for this counter are centralized in a single register called TCON.

A watchdog timer, called Timer I, is for use with the I^2C subsystem. In I^2C applications, this timer is dedicated to time-generation and bus monitoring of the I^2C. In non-I^2C applications, it is available for use as a fixed time-base.

Interrupt Subsystem — Fixed Priority

The IP register and the 2-level interrupt system of the SC80C51 are eliminated. Simultaneous interrupt conditions are resolved by a single-level, fixed priority as follows:

Highest priority:	Pin $\overline{INT0}$
	Counter/timer flag 0
	Pin $\overline{INT1}$
	PWM
	Timer I
	Serial I^2C
Lowest priority:	ADC

Serial Communications

The 8XC752 contains an I^2C serial communications port instead of the 80C51 UART. The I^2C serial port is a single bit hardware interface with all of the hardware necessary to support multimaster and slave operations. Also included are receiver digital filters and timer (timer I) for communication watch-dog purposes. The I^2C serial port is controlled through four special function registers; I^2C control, I^2C data, I^2C status, and I^2C configuration.

Pulse Width Modulation Output (P0.4)

The PWM outputs pulses of programmable length and interval. The repetition frequency is defined by an 8-bit prescaler which generates the clock for the counter. The prescaler register is PWMP. The prescaler and counter are not associated with any other timer. The 8-bit counter counts modulo 255, that is from 0 to 254 inclusive. The value of the 8-bit counter is compared to the contents of a compare register, PWM. When the counter value matches the contents of this register, the output of the PWM is set high. When the counter reaches zero, the output of the PWM is set low. The pulse width ratio (duty cycle) is defined by the contents of the compare register and is in the range of 0 to 1 programmed in increments of 1/255. The PWM output can be set to be continuously high by loading the compare register with 0 and the output can be set to be continuously low by loading the compare register with 255. The PWM output is enabled by a bit in a special function register, PWENA. When enabled, the pin output is driven with a fully active pull-up. That is, when the output is high, a strong pull-up is continuously applied. when disabled, the pin functions as a normal bidirectional I/O pin, however, the counter remains active.

The PWM function is disabled during RESET and remains disabled after reset is removed until re-enabled by software. The PWM output is high during power down and idle. The counter is disabled during idle. The repetition frequency of the PWM is given by:

$$f_{PWM} = f_{OSC} / 2 (1 + PWMP) 255$$

The low/high ratio of the PWM signal is PWM / (255 – PWM) for PWM not equal to 255. For PWM = 255, the output is always low.

The repetition frequency range is 92Hz to 23.5kHz for an oscillator frequency of 12MHz.

An interrupt will be asserted upon PWM counter overflow if the interrupt is not masked off.

The PWM output is an alternative function of P0.4. In order to use this port as a bidirectional I/O port, the PWM output must be disabled by clearing the enable/disable bit in PWENA. In this case, the PWM subsystem can be used as an interval timer by enabling the PWM interrupt.

A/D Converter

The analog input circuitry consists of a 5-input analog multiplexer and an A to D converter with 8-bit resolution. The conversion takes 40 machine cycles, i.e., 40µs at 12MHz oscillator frequency. The A/D converter is controlled using the ADCON control register. Input channels are selected by the analog multiplexer through ADCON register bits 0–2.

Special Function Register Addresses

Special function registers for the 8XC752 are identical to those of the SC80C51, except for the changes listed below:

SC80C51 special function registers not present in the 8XC752 are TMOD (89), P2 (A0) and IP (B8). The SC80C51 registers TH1, TL1, SCON, and SBUF are replaced with the 8XC752 registers RTH, RTL, I2CON, and I2DAT, respectively. Additional special function registers are I2CFG (D8) and I2STA (FB), ADCON (A0), ADAT (84), PWM (8E), PWMP (8F), and PWENA (FE). See Table 2.

Table 2. I^2C Special Function Register Addresses

REGISTER ADDRESS			BIT ADDRESS							
NAME	SYMBOL	ADDRESS	MSB							LSB
I^2C control	I2CON	98	9F	9E	9D	9C	9B	9A	99	98
I^2C data	I2DAT	99	–	–	–	–	–	–	–	–
I^2C configuration	I2CFG	D8	DF	DE	DD	DC	DB	DA	D9	D8
I^2C status	I2STA	F8	FF	FE	FD	FC	FB	FA	F9	F8

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage from V_{CC} to V_{SS}	–0.5 to +6.5	V
Voltage from any pin to V_{SS} (except V_{PP})	–0.5 to V_{CC} + 0.5	V
Power dissipation	1.0	W
Voltage from V_{PP} pin to V_{SS}	–0.5 to + 13.0	V

DC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or –40°C to +85°C, AV_{CC} = 5V ±5, AV_{SS} = 0V[3]
83C752: V_{CC} = 5V ± 20%, 87C752: V_{CC} = 5V ± 10%, VSS = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS[3] MIN	TYPICAL[1]	MAX	UNIT
I_{CC}	Supply current (see Figure 3)					
Inputs						
V_{IL}	Input low voltage, except SDA, SCL		–0.5		$0.2V_{CC}$–0.1	V
V_{IH}	Input high voltage, except X1, RST		$0.2V_{CC}$+0.9		V_{CC}+0.5	V
V_{IH1}	Input high voltage, X1, RST		$0.7V_{CC}$		V_{CC}+0.5	V
	SDA, SCL:					
V_{IL1}	Input low voltage		–0.5		$0.3V_{CC}$	V
V_{IH2}	Input high voltage		$0.7V_{CC}$		V_{CC}+0.5	V
Outputs						
V_{OL}	Output low voltage, ports 1, 3, 0.3, and 0.4 (PWM disabled)	I_{OL} = 1.6mA			0.45	V
V_{OL1}	Output low voltage, port 0.2	I_{OL} = 3.2mA			0.45	V
V_{OH}	Output high voltage, ports 1, 3, 0.3, and 0.4 (PWM disabled)	I_{OH} = –60µA,	2.4			V
		I_{OH} = –25µA	$0.75V_{CC}$			V
		I_{OH} = –10µA	$0.9V_{CC}$			V
		I_{OH} = –400µA	2.4			V
V_{OH2}	Output high voltage, 0.4 (PWM enabled)	I_{OH} = –40µA	$0.9V_{CC}$			V
	Port 0.0 and 0.1 (I^2C) – Drivers					
V_{OL2}	Output low voltage	I_{OL} = 3mA (over V_{CC} range)			0.4	V
	Driver, receiver combined:					
C	Capacitance				10	pF
I_{IL}	Logical 0 input current, ports 1, 3, 0.3, and 0.4 (PWM disabled)[10]	V_{IN} = 0.45V			–50	µA
I_{TL}	Logical 1 to 0 transition current, ports 1, 3, 0.3 and 0.4[10]	V_{IN} = 2V			–650	µA
I_{LI}	Input leakage current, port 0.0, 0.1 and 0.2	0.45 < V_{IN} < V_{CC}			±10	µA
R_{RST}	Reset pull-down resistor		25		175	kΩ
C_{IO}	Pin capacitance	Test freq = 1MHz, T_{amb} = 25°C			10	pF
I_{PD}	Power-down current[4]	V_{CC} = 2 to 5.5V V_{CC} = 2 to 6.0V (83C752)			50	µA
V_{PP}	V_{PP} program voltage (87C752 only)	V_{SS} = 0V V_{CC} = 5V±10% T_{amb} = 21°C to 27°C	12.5		13.0	V
I_{PP}	Program current (87C752 only)	V_{PP} = 13.0V			50	mA
Analog Inputs (A/D guaranteed only with quartz window covered).						
AV_{CC}	Analog supply voltage[9]	AV_{CC} = V_{CC}±0.2V	4.5		5.5	V
AI_{CC}	Analog operating supply current	AV_{CC} = 5.12V			3[8]	mA
AV_{IN}	Analog input voltage		AV_{SS}–0.2		AV_{CC}+0.2	V
C_{IA}	Analog input capacitance				15	pF

DC ELECTRICAL CHARACTERISTICS (Continued)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS[3] MIN	TYPICAL[1]	MAX	UNIT
Analog Inputs (Continued) (A/D guaranteed only with quartz window covered).						
t_{ADS}	Sampling time				$8t_{CY}$	s
t_{ADC}	Conversion time				$40t_{CY}$	s
R	Resolution				8	bits
E_{RA}	Relative accuracy				±1	LSB
OS_e	Zero scale offset				±1	LSB
G_e	Full scale gain error				0.4	%
M_{CTC}	Channel to channel matching				±1	LSB
C_t	Crosstalk	0–100kHz			–60	dB

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.
4. Power-down I_{CC} is measured with all output pins disconnected; port 0 = V_{CC}; X2, X1 n.c.; RST = V_{SS}.
5. I_{CC} is measured with all output pins disconnected; X1 driven with t_{CLCH}, t_{CHCL} = 5ns, V_{IL} = V_{SS} + 0.5V, V_{IH} = V_{CC} – 0.5V; X2 n.c.; RST = port 0 = V_{CC}. I_{CC} will be slightly higher if a crystal oscillator is used.
6. Idle I_{CC} is measured with all output pins disconnected; X1 driven with t_{CLCH}, t_{CHCL} = 5ns, V_{IL} = V_{SS} + 0.5V, V_{IH} = V_{CC} – 0.5V; X2 n.c.; port 0 = V_{CC}; RST = V_{SS}.
7. Load capacitance for ports = 80pF.
8. The resistor ladder network is not disconnected in the power down or idle modes. Thus, to conserve power, the user may remove AV_{CC}.
9. If the A/D function is not required, or if the A/D function is only needed periodically, AV_{CC} may be removed without affecting the operation of the digital circuitry. Contents of ADCON and ADAT are not guaranteed to be valid. If AV_{CC} is removed, the A/D inputs must be lowered to less than 0.5V. Digital inputs on P1.0–P1.4 will not function normally.
10. These parameters do not apply to P1.0–P1.4 if the A/D function is enabled.

A/D CONVERTER PARAMETER DEFINITIONS

The following definitions are included to clarify some specifications given and do not represent a complete set of A/D parameter definitions.

Absolute Accuracy Error

Absolute accuracy error of a given output is the difference between the theoretical analog input voltage to produce a given output and the actual analog input voltage required to produce the same code. Since the same output code is produced by a band of input voltages, the "required input voltage" is defined as the midpoint of the band of input voltage that will produce that code. Absolute accuracy error not specified with a code is the maximum over all codes.

Nonlinearity

If a straight line is drawn between the end points of the actual converter characteristics such that zero offset and full scale errors are removed, then non-linearity is the maximum deviation of the code transitions of the actual characteristics from that of the straight line so constructed. This is also referred to as relative accuracy and also integral non-linearity.

Differential Non-Linearity

Differential non-linearity is the maximum difference between the actual and ideal code widths fo the converter. The code widths are the differences expressed in LSB between the code transition points, as the input voltage is varied through the range for the complete set of codes.

Gain Error

Gain error is the deviation between the ideal and actual analog input voltage required to cause the final code transition to a full-scale output code after the offset error has been removed. This may sometimes be referred to as full scale error.

Offset Error

Offset error is the difference between the actual input voltage that causes the first code transition and the ideal value to cause the first code transition. This ideal value is 1/2 LSB above V_{ref-}.

Channel to Channel Matching

Channel to channel matching is the maximum difference between the corresponding code transitions of the actual characteristics taken from different channels under the same temperature, voltage and frequency conditions.

Crosstalk

Crosstalk is the measured level of a signal at the output of the converter resulting from a signal applied to one deselected channel.

Total Error

Maximum deviation of any step point from a line connecting the ideal first transition point to the ideal last transition point.

Relative Accuracy

Relative accuracy error is the deviation of the ADC's actual code transition points from the ideal code transition points on a straight line which connects the ideal first code transition point and the final code transition point, after nulling offset error and gain error. It is generally expressed in LSBs or in percent of FSR.

AC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10% (87C752), V_{CC} = 5V ±20% (83C752), V_{SS} = 0V[3, 7]

SYMBOL	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
		MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	Oscillator frequency:			3.5 3.5 0.5	12 16 12	MHz MHz MHz
External Clock (Figure 1)						
t_{CHCX}	High time	20		20		ns
t_{CLCX}	Low time	20		20		ns
t_{CLCH}	Rise time		20		20	ns
t_{CHCL}	Fall time		20		20	ns

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

C – Clock
D – Input data
H – Logic level high
L – Logic level low
Q – Output data
T – Time
V – Valid
X – No longer a valid logic level
Z – Float

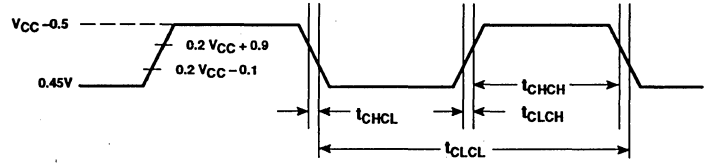


Figure 1. External Clock Drive

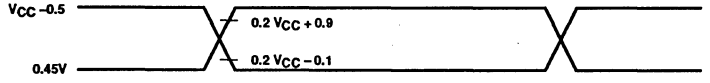


Figure 2. AC Testing Input/Output

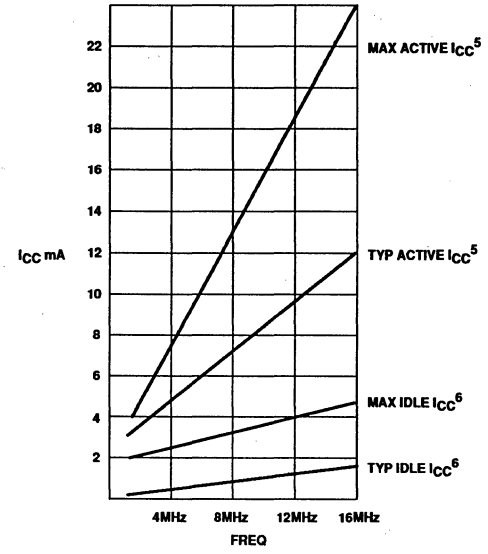


Figure 3. I_{CC} vs. FREQ
Maximum I_{CC} values taken at V_{CC} = 5.5V and worst case temperature.
Typical I_{CC} values taken at V_{CC} = 5.0V and 25°C.
Notes 5 and 6 refer to AC Electrcial Characteristics.

PROGRAMMING CONSIDERATIONS

EPROM Characteristics

The 87C752 is programmed by using a modified Quick-Pulse Programming algorithm similar to that used for devices such as the 87C451 and 87C51. It differs from these devices in that a serial data stream is used to place the 87C752 in the programming mode.

Figure 4 shows a block diagram of the programming configuration for the 87C752. Port pin P0.2 is used as the programming voltage supply input (V_{PP} signal). Port pin P0.1 is used as the program (PGM/) signal. This pin is used for the 25 programming pulses.

Port 3 is used as the address input for the byte to be programmed and accepts both the high and low components of the eleven bit address. Multiplexing of these address components is performed using the ASEL input. The user should drive the ASEL input high and then drive port 3 with the high order bits of the address. ASEL should remain high for at least 13 clock cycles. ASEL may then be driven low which latches the high order bits of the address internally. The high address should remain on port 3 for at least two clock cycles after ASEL is driven low. Port 3 may then be driven with the low byte of the address. The low address will be internally stable 13 clock cycles later. The address will remain stable provided that the low byte placed on port 3 is held stable and ASEL is kept low. **Note:** ASEL needs to be pulsed high only to change the high byte of the address.

Port 1 is used as a bidirectional data bus during programming and verify operations. During programming mode, it accepts the byte to be programmed. During verify mode, it provides the contents of the EPROM location specified by the address which has been supplied to Port 3.

The XTAL1 pin is the oscillator input and receives the master system clock. This clock should be between 1.2 and 6MHz.

The RESET pin is used to accept the serial data stream that places the 87C752 into various programming modes. This pattern consists of a 10-bit code with the LSB sent first. Each bit is synchronized to the clock input, X1.

Programming Operation

Figures 5 and 6 show the timing diagrams for the program/verify cycle. RESET should initially be held high for at least two machine cycles. P0.1 (PGM/) and P0.2 (V_{PP}) will be at V_{OH} as a result of the RESET operation. At this point, these pins function as normal quasi-bidirectional I/O ports and the programming equipment may pull these lines low. However, prior to sending the 10-bit code on the RESET pin, the programming equipment should drive these pins high (V_{IH}). The RESET pin may now be used as the serial data input for the data stream which places the 87C752 in the programming mode. Data bits are sampled during the clock high time and thus should only change during the time that the clock is low. Following transmission of the last data bit, the RESET pin should be held low.

Next the address information for the location to be programmed is placed on port 3 and ASEL is used to perform the address multiplexing, as previously described. At this time, port 1 functions as an output.

A high voltage V_{PP} level is then applied to the V_{PP} input (P0.2). (This sets Port 1 as an input port). The data to be programmed into the EPROM array is then placed on Port 1. This is followed by a series of programming pulses applied to the PGM/ pin (P0.1). These pulses are created by driving P0.1 low and then high. This pulse is repeated until a total of 25 programming pulses have occurred. At the conclusion of the last pulse, the PGM/ signal should remain high.

The V_{PP} signal may now be driven to the V_{OH} level, placing the 87C752 in the verify mode. (Port 1 is now used as an output port). After four machine cycles (48 clock periods), the contents of the addressed location in the EPROM array will appear on Port 1.

The next programming cycle may now be initiated by placing the address information at the inputs of the multiplexed buffers, driving the V_{PP} pin to the V_{PP} voltage level, providing the byte to be programmed to Port1 and issuing the 26 programming pulses on the PGM/ pin, bringing V_{PP} back down to the V_C level and verifying the byte.

Programming Modes

The 87C752 has four programming features incorporated within its EPROM array. These include the USER EPROM for storage of the application's code, a 16-byte encryption key array and two security bits. Programming and verification of these four elements are selected by a combination of the serial data stream applied to the RESET pin and the voltage levels applied to port pins P0.1 and P0.2. The various combinations are shown in Table 3.

Encryption Key Table

The 87C752 includes a 16-byte EPROM array that is programmable by the end user. The contents of this array can then be used to encrypt the program memory contents during a program memory verify operation. When a program memory verify operation is performed, the contents of the program memory location is XNOR'ed with one of the bytes in the 16-byte encryption table. The resulting data pattern is then provided to port 1 as the verify data. The encryption mechanism can be disable, in essence, by leaving the bytes in the encryption table in their erased state (FFH) since the XNOR product of a bit with a logical one will result in the original bit. The encryption bytes are mapped with the code memory in 16-byte groups. the first byte in code memory will be encrypted with the first byte in the encryption table; the second byte in code memory will be encrypted with the second byte in the encryption table and so forth up to and including the 16the byte. The encryption repeats in 16-byte groups; the 17th byte in the code memory will be encrypted with the first byte in the encryption table, and so forth.

Security Bits

Two security bits, security bit 1 and security bit 2, are provided to limit access to the USER EPROM and encryption key arrays. Security bit 1 is the program inhibit bit, and once programmed performs the following functions:

1. Additional programming of the USER EPROM is inhibited.
2. Additional programming of the encryption key is inhibited.
3. Verification of the encryption key is inhibited.
4. Verification of the USER EPROM and the security bit levels may still be performed.

(If the encryption key array is being used, this security bit should be programmed by the user to prevent unauthorized parties from reprogramming the encryption key to all logical zero bits. Such programming would provide data during a verify cycle that is the logical complement of the USER EPROM contents).

Security bit 2, the verify inhibit bit, prevents verification of both the USER EPROM array and the encryption key arrays. The security bit levels may still be verified.

Programming and Verifying Security Bits

Security bits are programmed employing the same techniques used to program the USER EPROM and KEY arrays using serial data streams and logic levels on port pins indicated in Table 3. When programming either security bit, it is not necessary to provide address or data information to the 87C752 on ports 1 and 3.

Verification occurs in a similar manner using the RESET serial stream shown in Table 3. Port 3 is not required to be driven and the results of the verify operation will appear on ports 1.6 and 1.7.

Ports 1.7 contains the security bit 1 data and is a logical one if programmed and a logical zero if erased. Likewise, P1.6 contains the security bit 2 data and is a logical one if programmed and a logical zero if erased.

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. **For this and secondary effects, it is recommended that an opaque label be placed over the window.** For elevated temperature or environments where solvents are being used, apply Kapton tape Flourless part number 2345–5 or equivalent.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 angstroms) to an integrated dose of at least 15W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000uW/cm^2 rating for 20 to 39 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

Table 3. Implementing Program/Verify Modes

OPERATION	SERIAL CODE	P0.1 (PGM/)	P0.2 (V_{PP})
Program user EPROM	296H	–*	V_{PP}
Verify user EPROM	296H	V_{IH}	V_{IH}
Program key EPROM	292H	–*	V_{PP}
Verify key EPROM	292H	V_{IH}	V_{IH}
Program security bit 1	29AH	–*	V_{PP}
Program security bit 2	298H	–*	V_{PP}
Verify security bits	29AH	V_{IH}	V_{IH}

NOTE:
* Pulsed from V_{IH} to V_{IL} and returned to V_{IH}.

EPROM PROGRAMMING AND VERIFICATION

T_{amb} = 21°C to +27°C, V_{CC} = 5V ±10%, V_{SS} = 0V

SYMBOL	PARAMETER	MIN	MAX	UNIT
$1/t_{CLCL}$	Oscillator/clock frequency	1.2	6	MHz
t_{AVGL}*	Address setup to P0.1 (PROG–) low	10µs + 24t_{CLCL}		
t_{GHAX}	Address hold after P0.1 (PROG–) high	48t_{CLCL}		
t_{DVGL}	Data setup to P0.1 (PROG–) low	38t_{CLCL}		
t_{DVGL}	Data setup to P0.1 (PROG–) low	38t_{CLCL}		
t_{GHDX}	Data hold after P0.1 (PROG–) high	36t_{CLCL}		
t_{SHGL}	V_{PP} setup to P0.1 (PROG–) low	10		µs
t_{GHSL}	V_{PP} hold after P0.1 (PROG–)	10		µs
t_{GLGH}	P0.1 (PROG–) width	90	110	µs
t_{AVQV}**	V_{PP} low (V_{CC}) to data valid		48t_{CLCL}	
t_{GHGL}	P0.1 (PROG–) high to P0.1 (PROG–) low	10		µs
t_{SYNL}	P0.0 (sync pulse) low	4t_{CLCL}		
t_{SYNH}	P0.0 (sync pulse) high	8t_{CLCL}		
t_{MASEL}	ASEL high time	13t_{CLCL}		
t_{MAHLD}	Address hold time	2t_{CLCL}		
t_{HASET}	Address setup to ASEL	13t_{CLCL}		
t_{ADSTA}	Low address to address stable	13t_{CLCL}		

NOTES:
* Address should be valid at least 24t_{CLCL} before the rising edge of P0.2 (V_{PP}).
** For a pure verify mode, i.e., no program mode in between, t_{AVQV} is 14t_{CLCL} maximum.

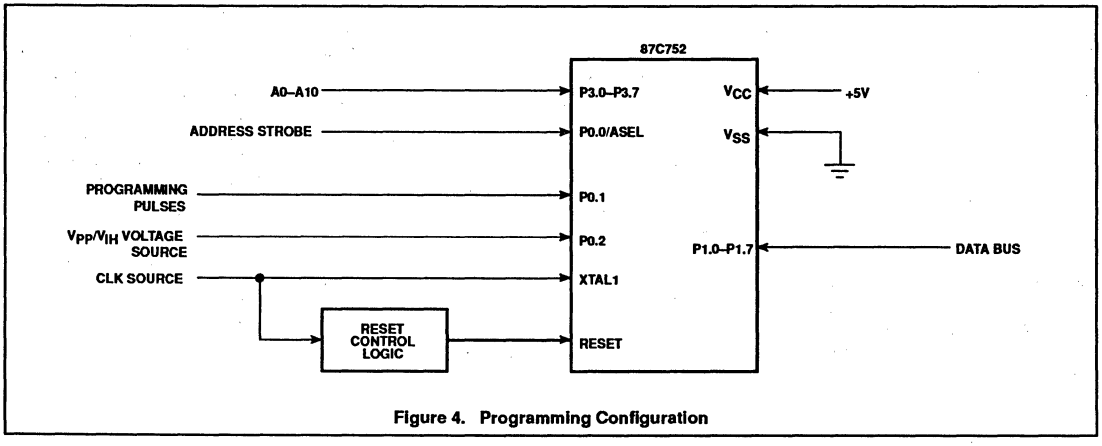


Figure 4. Programming Configuration

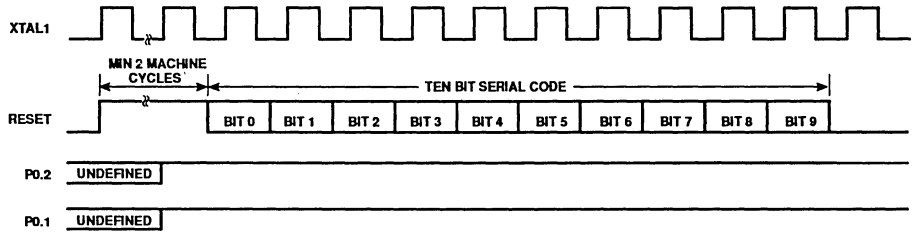


Figure 5. Entry into Program/Verify Modes

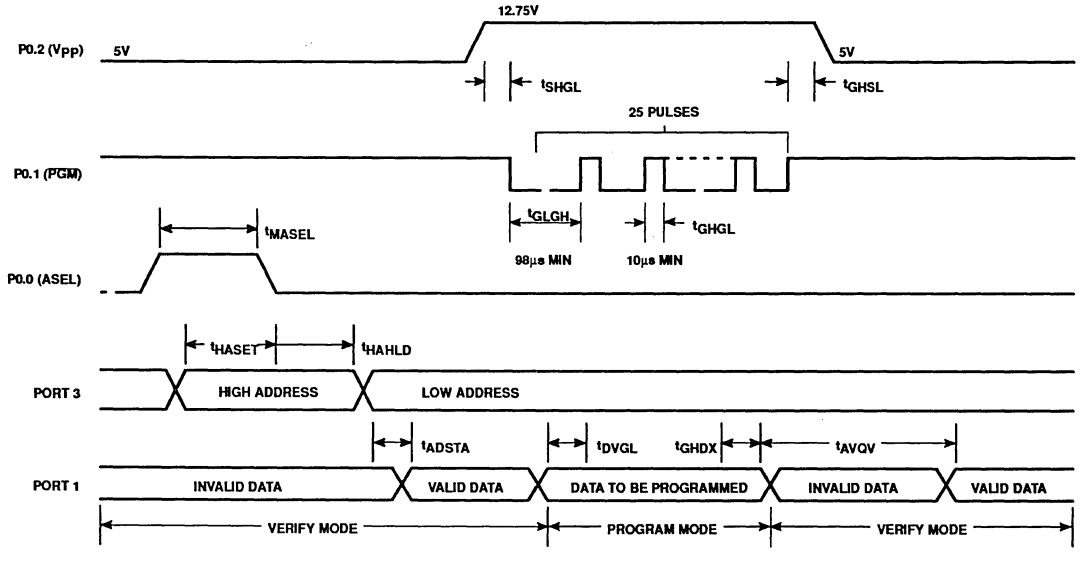


Figure 6. Program/Verify Cycle

Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specifications defined by Philips.

8XC851 OVERVIEW

The 80C851 and the 83C851 (hereafter referred to collectively as the 8XC851) are EEPROM expanded versions of the 80C51. These devices are pin-for-pin compatible with the 80C51 with the addition of 256 bytes of EEPROM. The addition of the EEPROM makes these devices suitable for a variety of applications, specifically control and security systems.

The 8XC851 includes a 4k × 8 ROM, a 128 × 8 RAM, a 256 × 8 electrically erasable programmable read-only memory (EEPROM), 32 I/O lines, two 16-bit timer/counters, a seven-source, two priority level nested interrupt structure, a serial I/O port for either full duplex UART or I/O expansion, and an on-chip oscillator and clock circuit. The 80C851 includes all of the 83C851 features except the on-board 4k × 8 ROM.

The 8XC851 has two software selectable modes of reduced activity for further power reduction: idle mode and power-down mode. Idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. Power-down mode freezes the oscillator, causing all other chip functions to be inoperative while maintaining the RAM contents.

In addition, some special security features are implemented:

- ROM code protection:
  Mask-programmable. When implemented, access to the internal ROM is possible only when executing internal program memory; it is not possible to access the internal ROM when executing external program memory.
- EEPROM protection:
  In the security mode (enabled when the security bit is set), the contents of the EEPROM are protected and, when executing external program memory, no read or write operation to the EEPROM is possible except "Blockerase" ("Blockerase" clears all bytes, including the byte containing the security bits).

The 8XC851 features include:

- 80C51 pin-for-pin compatibility
- 4k × 8 ROM
- 128 × 8 RAM
- 256 × 8 EEPROM
  - On-chip voltage multiplier for erase/write
  - 50,000 erase/write cycles per byte
  - 10 years non-volatile data retention
  - Infinite number of read cycles
- Two 16-bit counter/timers
- Two external interrupts
- External memory addressing capability
  - 64k ROM and 64k RAM
- Low power consumption
  - Idle mode
  - Power-down mode
- ROM code protection
- EEPROM security mode

Differences from the 80C51

Special Function Registers

The SFRs are identical to those of the standard 80C51 with the exception of five registers (EADRL, EADRH, EDAT, ECNTRL, and ETIM) that have been added to allow control of the 256 bytes of EEPROM. Table 1 is a detailed expansion of the special function registers.

Refer to the 80C851 data sheet for additional information.

Table 1. 8XC851 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
			EF	EE	ED	EC	EB	EA	E9	E8	
DPTR:	Data pointer (2 bytes):										
DPH	High byte	83H									00H
DPL	Low byte	82H									00H
EADRH#	EEPROM addr reg-high	F3H									80H
EADRL#	EEPROM addr reg-low	F2H									00H
ECNTRL#	EEPROM control reg	F6H	IFE	EEINT	EWP	–	ECNTR L3	ECNTR L2	ECNTR L1	ECNTR L0	00H
EDAT#	EEPROM data register	F4H									xxH
ETIM#	EEPROM timer register	F5H									08H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt priority	B8H	–	–	–	PS	PT1	PX1	PT0	PX0	xxx00000B
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt enable	A8H	EA	–	–	ES	ET1	EX1	ET0	EX0	0xx00000B
P0*	Port 0	80H	87	86	85	84	83	82	81	80	FFH
P1*	Port 1	90H	97	96	95	94	93	92	91	90	FFH
P2*	Port 2	A0H	A7	A6	A5	A4	A3	A2	A1	A0	FFH
P3*	Port 3	B0H	B7	B6	B5	B4	B3	B2	B1	B0	FFH
PCON	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
SBUF	Serial data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial port control	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	00H
TCON*	Timer/counter control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
TMOD	Timer/counter mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
TH0	Timer 0 high byte	8CH									00H
TH1	Timer 1 high byte	8DH									00H
TL0	Timer 0 low byte	8AH									00H
TL1	Timer 1 low byte	8BH									00H

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.

CMOS single-chip 8-bit microcontroller with on-chip EEPROM

80C851/83C851

DESCRIPTION

The Philips 80C851/83C851 is a high-performance microcontroller fabricated with Philips high-density CMOS technology. The 80C851/83C851 has the same instruction set as the 80C51. The Philips CMOS technology combines the high speed and density characteristics of HMOS with the low power attributes of CMOS. The Philips epitaxial substrate minimizes latch-up sensitivity.

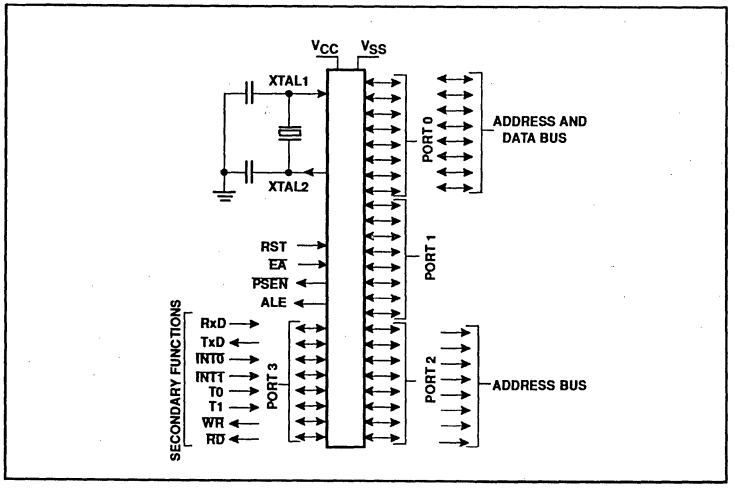
The 80C851/83C851 contains a 4k × 8 ROM with mask-programmable ROM code protection, a 128 × 8 RAM, 256 × 8 EEPROM, 32 I/O lines, two 16-bit counter/timers, a seven-source, five vector, two-priority level nested interrupt structure, a serial I/O port for either multi-processor communications, I/O expansion or full duplex UART, and on-chip oscillator and clock circuits.

In addition, the 80C851/83C851 has two software selectable modes of power reduction — idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. The power-down mode saves the RAM and EEPROM contents but freezes the oscillator, causing all other chip functions to be inoperative.

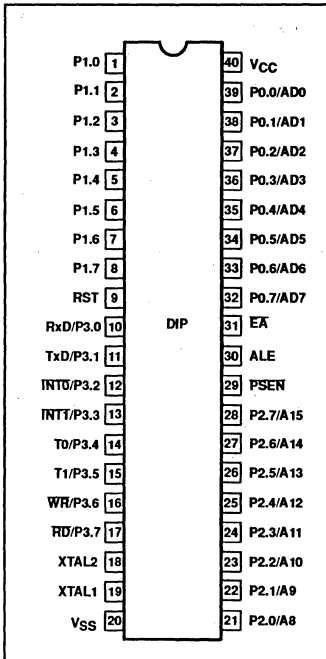
FEATURES

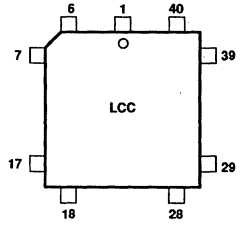
- 80C51 based architecture
  - 4k × 8 ROM
  - 128 × 8 RAM
  - Two 16-bit counter/timers
  - Full duplex serial channel
  - Boolean processor
- Non-volatile 256 × 8-bit EEPROM (electrically erasable programmable read only memory)
  - On-chip voltage multiplier for erase/write
  - 50,000 erase/write cycles per byte
  - 10 years non-volatile data retention
  - Infinite number of read cycles
  - User selectable security mode
  - Block erase capability
- Mask-programmable ROM code protection
- Memory addressing capability
  - 64k ROM and 64k RAM
- Power control modes:
  - Idle mode
  - Power-down mode
- CMOS and TTL compatible
- 1.2 to 16MHz
- Three package styles
- Three temperature ranges

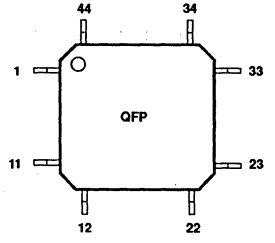
LOGIC SYMBOL



PIN CONFIGURATIONS







SEE PAGE 612 FOR QFP AND LCC PIN FUNCTIONS.

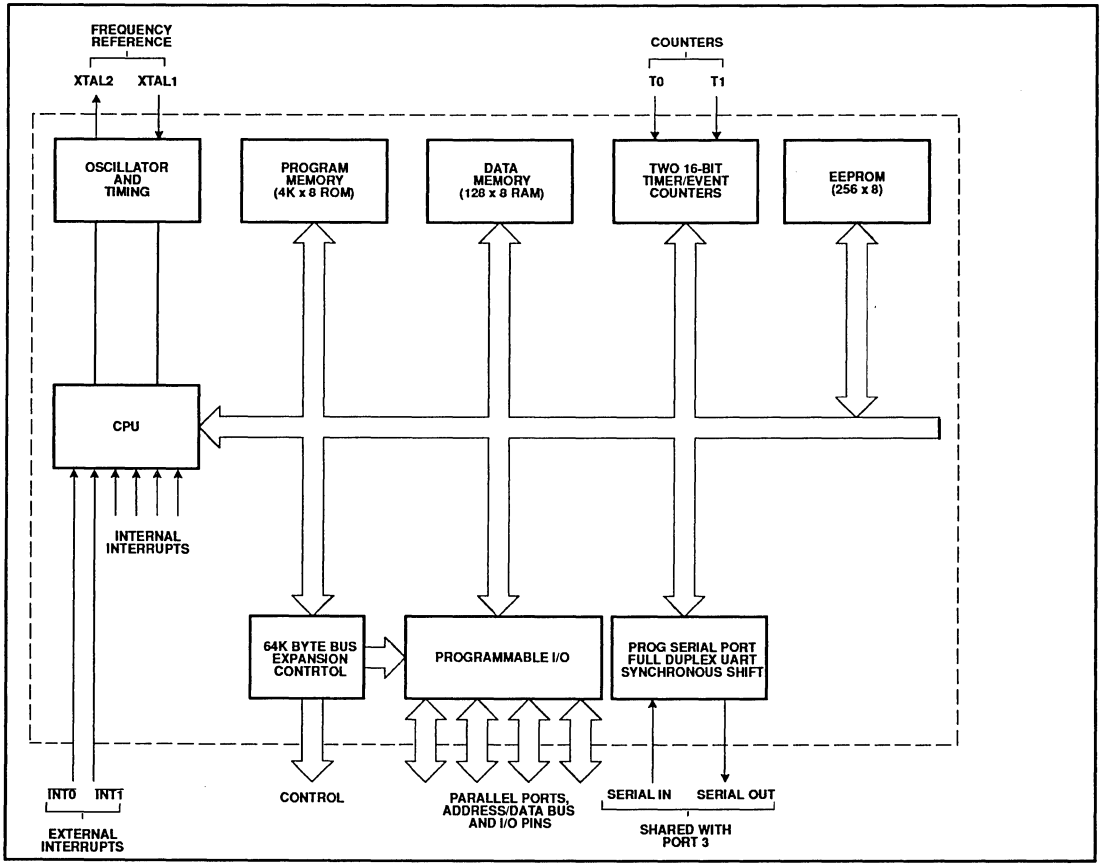
CMOS single-chip 8-bit microcontroller with on-chip EEPROM

80C851/83C851

PART NUMBER SELECTION

PHILIPS PART ORDER NUMBER PART MARKING		SIGNETICS PART ORDER NUMBER		TEMPERATURE AND PACKAGE	FREQUENCY (MHz)
ROMless Version	ROM Version	ROMless Version	ROM Version		
PCB80C851-2-16P	PCB83C851-2P	S80C851-4N40	S83C851-4N40	0 to +70°C, plastic DIP	1.2 to 16
PCB80C851-2-16WP	PCB83C851-2WP	S80C851-4A44	S83C851-4A44	0 to +70°C, plastic PLCC	1.2 to 16
PCB80C851-2-16H	PCB83C851-2H	S80C851-4B44	S83C851-4B44	0 to +70°C, plastic QFP	1.2 to 16
PCF80C851-2-16P	PCF83C851-2P	S80C851-5N40	S83C851-5N40	–40 to +85°C, plastic DIP	1.2 to 16
PCF80C851-2-16WP	PCF83C851-2WP	S80C851-5A44	S83C851-5A44	–40 to +85°C, plastic PLCC	1.2 to 16
PCF80C851-2-16H	PCF83C851-2H	S80C851-5B44	S83C851-5B44	–40 to +85°C, plastic QFP	1.2 to 16
PCA80C851-2-16P	PCA83C851-2P	S80C851-6N40	S83C851-6N40	–40 to +125°C, plastic DIP	1.2 to 16
PCA80C851-2-16WP	PCA83C851-2WP	S80C851-6A44	S83C851-6A44	–40 to +125°C, plastic PLCC	1.2 to 16
PCA80C851-2-16H	PCA83C851-2H	S80C851-6B44	S83C851-6B44	–40 to +125°C, plastic QFP	1.2 to 16

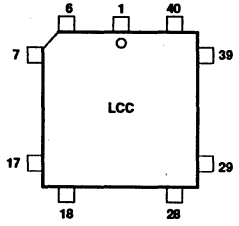
BLOCK DIAGRAM



CMOS single-chip 8-bit microcontroller with on-chip EEPROM

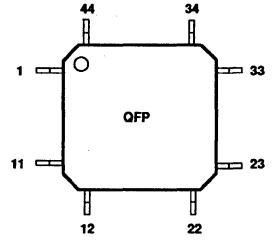
80C851/83C851

LCC PIN FUNCTIONS



Pin	Function	Pin	Function
1	NC	23	NC
2	P1.0	24	P2.0/A8
3	P1.1	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6	30	P2.6/A14
9	P1.7	31	P2.7/A15
10	RST	32	$\overline{PSEN}$
11	P3.0/RxD	33	ALE
12	NC	34	NC
13	P3.1/TxD	35	$\overline{EA}$
14	P3.2/$\overline{INT0}$	36	P0.7/AD7
15	P3.3/$\overline{INT1}$	37	P0.6/AD6
16	P3.4/T0	38	P0.5/AD5
17	P3.5/T1	39	P0.4/AD4
18	P3.6/$\overline{WR}$	40	P0.3/AD3
19	P3.7/$\overline{RD}$	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	V_{SS}	44	V_{CC}

QFP PIN FUNCTIONS



Pin	Function	Pin	Function
1	P1.5	23	P2.5/A13
2	P1.6	24	P2.6/A14
3	P1.7	25	P2.7/A15
4	RST	26	$\overline{PSEN}$
5	P3.0/RxD	27	ALE
6	NC	28	NC
7	P3.1/TxD	29	$\overline{EA}$
8	P3.2/$\overline{INT0}$	30	P0.7/AD7
9	P3.3/$\overline{INT1}$	31	P0.6/AD6
10	P3.4/T0	32	P0.5/AD5
11	P3.5/T1	33	P0.4/AD4
12	P3.6/$\overline{WR}$	34	P0.3/AD3
13	P3.7/$\overline{RD}$	35	P0.2/AD2
14	XTAL2	36	P0.1/AD1
15	XTAL1	37	P0.0/AD0
16	V_{SS}	38	V_{CC}
17	NC	39	V_{SS}
18	P2.0/A8	40	P1.0
19	P2.1/A9	41	P1.1
20	P2.2/A10	42	P1.2
21	P2.3/A11	43	P1.3
22	P2.4/A12	44	P1.4

PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{SS}	20	22	16, 39	I	**Ground:** 0V reference.
V_{CC}	40	44	38	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–36	37–30	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s.
P1.0–P1.7	1–8	2–9	40–44, 1–3	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}).
P2.0–P2.7	21–28	24–31	18–25	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	5, 7–13	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the SC80C51 family, as listed below:
	10	11	5	I	**RxD (P3.0):** Serial input port
	11	13	7	O	**TxD (P3.1):** Serial output port
	12	14	8	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	9	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	10	I	**T0 (P3.4):** Timer 0 external input
	15	17	11	I	**T1 (P3.5):** Timer 1 external input
	16	18	12	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	13	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	4	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE	30	33	27	I/O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory.
$\overline{PSEN}$	29	32	26	O	**Program Store Enable:** The read strobe to external program memory. When the device is executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$	31	35	29	I	**External Access Enable:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H and 0FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 0FFFH.
XTAL1	19	21	15	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	14	O	**Crystal 2:** Output from the inverting oscillator amplifier.

EEPROM

Communications between the CPU and the EEPROM is accomplished via 5 special function registers; 2 address registers (high and low byte), 1 data register for read and write operations, 1 control register, and 1 timer register to adapt the erase/write time to the clock frequency. All registers can be read and written. Figure 1 shows a block diagram of the CPU, the EEPROM and the interface.

Register and Functional Description

Address Register (EADRH, EADRL)
The lower byte contains the address of one of the 256 bytes. The higher byte (EADRH) is for future extensions and for addressing the security bits (see Security Facilities). The EADRH register address is F3H. The EADRL register address is F2H.

Data Register (EDAT)
This register is required for read and write operations and also for row/block erase. In write mode, its contents are written to the addressed byte (for "row erase" and "block erase" the contents are don't care). The write pulse starts all operations, except read. In read mode, EDAT contains the data of the addressed byte. The EDAT register address is F4H.

Timer Register (ETIM)
The timer register is required to adapt the erase/write time to the oscillator frequency. The user has to ensure that the erase or write (program) time is neither too short or too long. The ETIM register address is F5H. Table 1 contains the values which must be written to the ETIM register by software for various oscillator frequencies (the default value is 08H after RESET).

The general formula is:

5ms Write time:

$$\text{Value (decimal, to be rounded up)} = \frac{f_{XTAL1}\ [kHz]}{204.8} - 2$$

10ms Write time:

$$\text{Value (decimal)} = \frac{f_{XTAL1}\ [kHz]}{96} - 2$$

Control Register (ECNTRL)
See Figure 2 for a description of this register. The ECNTRL register address is F6H.

Table 1. Values for the Timer Register (ETIM)

f_{XTAL1}	VALUES FOR ETIM			
	5ms WRITE TIME		10ms WRITE TIME	
	HEX	DEC	HEX	DEC
1.0MHz	03	3	08	8
2.0MHz	08	8	13	19
3.0MHz	0D	13	1D	29
4.0MHz	12	18	28	40
5.0MHz	17	23	32	50
6.0MHz	1C	28	3C	60
7.0MHz	21	33	47	71
8.0MHz	26	38	51	81
9.0MHz	2A	42	5C	92
10.0MHz	2F	47	66	102
11.0MHz	34	52	71	113
12.0MHz	39	57	7B	123
13.0MHz	3E	62		
14.0MHz	43	67		
15.0MHz	48	72		
16.0MHz	4D	77		

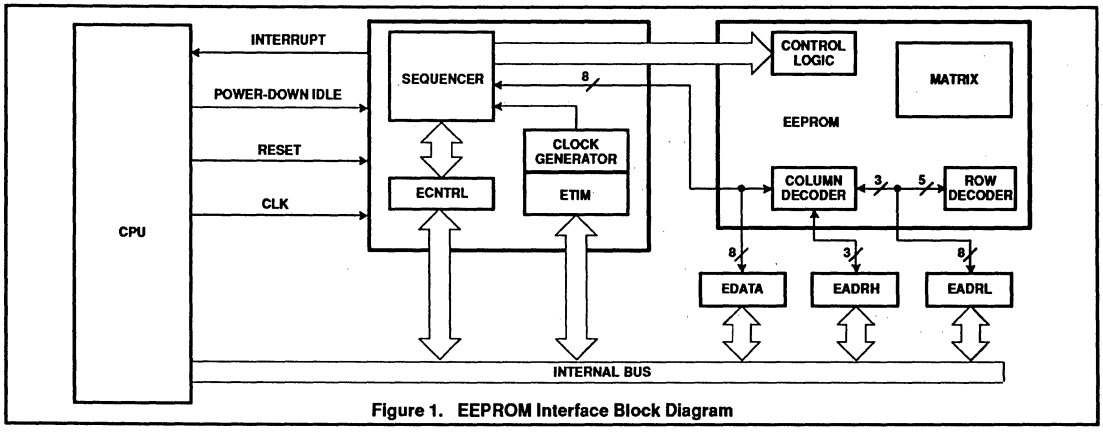


Figure 1. EEPROM Interface Block Diagram

7	6	5	4	3	2	1	0
IFE	EEINT	EWP	--	ECNTRL3	ECNTRL2	ECNTRL1	ECNTRL0

Bit	Symbol	Function
ECNTRL.7	IFE	Active high EEPROM interrupt flag: set by the sequencer or by software; reset by software. When set and enabled, this flag forces an interrupt to the same vector as the serial port interrupt (see Interrupt section).
ECNTRL.6	EEINT	EEPROM interrupt enable: set and reset by software (active high).
ECNTRL.5	EWP	Erase/write in progress flag: set and reset by the sequencer (active high). When EWP is set, access to the EEPROM is not possible. EWP cannot be set or reset by software.
ECNTRL.4		Reserved.
ECCTRL.3–ECNTRL.0		See table below.

Operation	ECNTRL.3	ECNTRL.2	ECNTRL.1	ECNTRL.0
Byte mode	0	0	0	0
Row erase	1	1	0	0
Page write*	–	–	–	–
Page erase/write*	–	–	–	–
block erase	1	0	1	0

*Future products.

Byte mode: Normal EEPROM mode, default mode after reset. In this mode, data can be read and written to one byte at a time.

Read mode: This is the default mode when byte mode is selected. This means that the contents of the addressed byte are available in the data register.

Write mode: This mode is activated by writing to the data register. The address register must be loaded first. Since the old contents are read first (by default), this allows the sequencer to decide whether an erase/write or write cycle only (data = 00H) is required.

Row erase: In this mode, the addressed row is cleared. The three LSBs of EADRL are not significant, i.e. the 8 bytes addressed by EADRL are cleared in the same time normally needed to clear one byte ($t_{ROWERASE} = t_E = t_W$). For the following write modes, only the write and not the erase/write cycle is required. For example, using the row erase mode, programming 8 bytes takes $t_{TOTOAL} = t_E + 8 \times t_W$ compared to $t_{TOTAL} = 8 \times t_E + 8 \times t_W$ ($t_E = t_{ERASE} \cdot t_W = t_{WRITE}$).

Page write: For future products.

Page erase/write: For future products

Block erase: In this mode all 256 bytes are cleared. The byte containing the security bits is also cleared. $t_{BLOCKERASE} = t_E$. The contents of EADRH, EADRL and EDAT are insignificant.

Program Sequences and Register Contents after Reset

The contents of the EEPROM registers after a Reset are the default values:

EADRH	= 1xxxxxxxB	(security bit address)
EADRL	= 00H	(security bit address)
ETIM	= 08H	(minimum erase time with the lowest permissible oscillator frequency)
ECNTRL	= 00H	(Byte mode, read)
EDAT	= xxH	(security bit)

```
Initialize:   MOV ETIM, ..
              MOV EADRH, ..

Read:         MOV EADRL, ..
              MOV .., EDAT

Write:        MOV EADRL, ..
              MOV EDAT, ..

Erase row:    MOV EADRL, ..          Row address. 3LSBs don't care
              MOV ECNTRL, #0CH       Erase row mode
              MOV EDAT, .. (EDAT) don't care

Erase block:  MOV ECNTRL, 0AH        Erase block mode
              MOV EDAT, .. (EDAT) don't care
```

If the security bit is to be altered, the program generally starts as follows:

```
MOV EADRH, #80H
MOV EADRL, #00H
```

Figure 2. Control Register (ECNTRL)

Security Facilities

EEPROM Protection

The EEPROM is protected using four security bits which are contained in an extra EEPROM byte at address 8000H (EADRH/EADRL). They can be set or cleared by software. To activate the EEPROM protection, the program sequence in byte mode must be as follows:

```
MOV  EADRH, #80H
MOV  EADRL, #00H
MOV  EDAT, #FFH
```

If two or more of these bits are reset, SB = 0, the security mode is disabled and the EEPROM is not protected. If three or four bits are set, SB = 1 and the $\overline{EA}$ mode differs from the internal access mode.

In this case, access to the EEPROM is only possible in one mode regardless of how the external access mode is reached (by pulling the $\overline{EA}$ pin low or by passing the 4K boundary). For SB = 1 and "external access" only, the "block erase" mode is enabled. The program sequence has to be as follows:

```
MOV  EADRH, #80H (security byte address)
MOV  EADRL, #00H (security byte address)
MOV  ECNTRL, #0AH (block erase mode)
MOV  EDAT, #xxH    (start block erase)
```

All 256 data bytes, the security bits, and SB will be cleared after completing this mode (EWP = 0). SB will also be affected in byte mode when writing to the security byte (not for SB = 1 and "external access"). Figure 3 illustrates the access to SB.

ROM Code Protection

Since the external access mode can only be selected by pulling the $\overline{EA}$ pin low during reset, it is not possible to read the internal program memory using the MOVC instruction while executing external program memory. Furthermore, it is not possible to change this mode to internal access within the MOVC cycle.

Additionally, a mask-programmable ROM code protection facility is available. When the program memory passes the 4K boundary using both the internal and external ROMs, it is not possible to access the internal ROM from the external program memory if the mask-programmable ROM security bit is set. An access to the lower 4K bytes of program memory using the MOVC instruction is only possible while executing internal program memory.

Also the verification mode (test-mode which writes the ROM contents to a port for comparison with a reference code) is not implemented for security reasons. A different test-mode is implemented for test purposes. This mode allows every bit to be tested. However, the internal code cannot be accessed via a port.

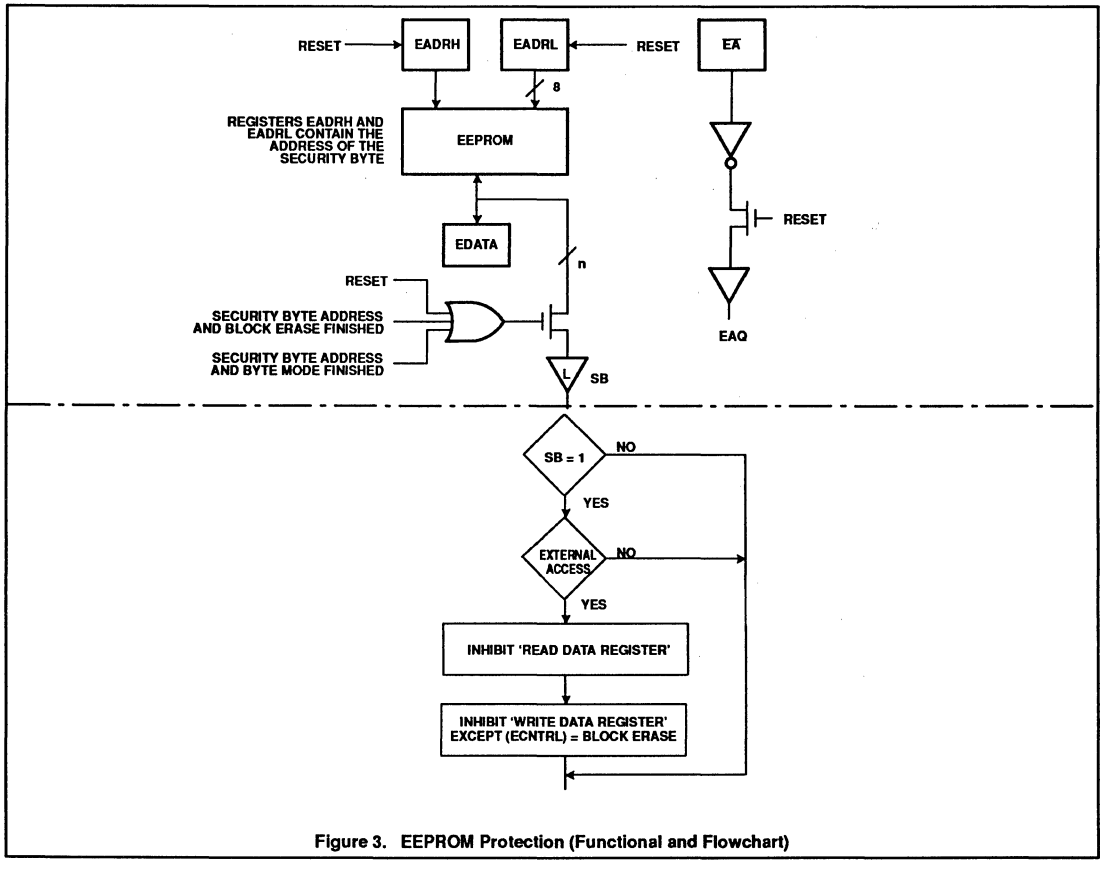


Figure 3. EEPROM Protection (Functional and Flowchart)

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 610.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-up reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-up, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

Note: Before entering the idle or power-down modes, the user has to ensure that there is no EEPROM erase/write cycle in progress (i.e., the EWP bit has to be reset before activating the idle or power-down modes; otherwise EEPROM accesses will be aborted).

IDLE MODE

In idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM and EEPROM are preserved. A hardware reset is the only way to terminate the power-down mode. The control bits for the reduced power modes are in the special function register PCON. Table 2 shows the state of the I/O ports during low current operating modes.

INTERRUPT SYSTEM

External events and the real-time-driven on-chip peripherals require service by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution, a multiple-source, two-priority-level, nested interrupt system is provided. Interrupt response latency is from 3µs to 7µs when using a 12MHz crystal. The S83C851 acknowledges interrupt requests from 7 sources as follows:

- $\overline{INT0}$ and $\overline{INT1}$: externally via pins 12 and 13, respectively,
- Timer 0 and timer 1: from the two internal counters,
- Serial port: from the internal serial I/O port or EEPROM (1 vector).

Each interrupt vectors to a separate location in program memory for its service program. Each source can be individually enabled (the EEPROM interrupt can only be enabled when the serial port interrupt is enabled) or disabled and can be programmed to a high or low priority level. All enabled sources can also be globally disabled or enabled. Both external interrupts can be programmed to be level-activated and are active low to allow "wire-ORing" of several interrupt sources to one input pin.

Note: The serial port and EEPROM interrupt flags must be cleared by software; all other flags are cleared by hardware.

Table 2. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	–65 to +150	°C
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Input or output DC current on any single I/O pin	±5	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.0	W

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C (V_{CC} = 5V ±20%), –40°C to +85°C (V_{CC} = 5V ±20%), or –40°C to +125°C (V_{CC} = 5V ±10%), V_{SS} = 0V

SYMBOL	PARAMETER	PART TYPE	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
V_{IL}	Input low voltage, except $\overline{EA}$	0 to +70°C –40 to +85°C –40 to +125°C		–0.5 –0.5 –0.5	$0.2V_{CC}$–0.1 $0.2V_{CC}$–0.15 $0.2V_{CC}$–0.25	V V V
V_{IL1}	Input low voltage to $\overline{EA}$	0 to +70°C –40 to +85°C –40 to +125°C		–0.5 –0.5 –0.5	$0.2V_{CC}$–0.3 $0.2V_{CC}$–0.35 $0.2V_{CC}$–0.45	V V V
V_{IH}	Input high voltage, except XTAL1, RST	0 to +70°C –40 to +85°C –40 to +125°C		$0.2V_{CC}$+0.9 $0.2V_{CC}$+1.0 $0.2V_{CC}$+1.0	V_{CC}+0.5 V_{CC}+0.5 V_{CC}+0.5	V V V
V_{IH1}	Input high voltage, XTAL1, RST	0 to +70°C –40 to +85°C –40 to +125°C		$0.7V_{CC}$ $0.7V_{CC}$+0.1 $0.7V_{CC}$+0.1	V_{CC}+0.5 V_{CC}+0.5 V_{CC}+0.5	
V_{OL}	Output low voltage, ports 1, 2, 3 [6]		I_{OL} = 1.6mA[4]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$ [6]		I_{OL} = 3.2mA[4]		0.45	V
V_{OH}	Output high voltage, ports 1, 2, 3, ALE, $\overline{PSEN}$		I_{OH} = –60µA, I_{OH} = –25µA, I_{OH} = –10µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$		V V V
V_{OH1}	Output high voltage, port 0 in external bus mode[5]		I_{OH} = –800µA, I_{OH} = –300µA, I_{OH} = –80µA	2.4 $0.75V_{CC}$ $0.9V_{CC}$		V V V
I_{IL}	Logical 0 input current, ports 1, 2, 3	0 to +70°C –40 to +85°C –40 to +125°C	V_{IN} = 0.45V		–50 –75 –75	µA µA µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3	0 to +70°C –40 to +85°C –40 to +125°C	V_{IN} = 2.0V		–650 –750 –750	µA µA µA
I_{L1}	Input leakage current, port 0, $\overline{EA}$		0.45V<V_i<V_{CC}		±10	µA
I_{CC}	Power supply current: Active mode @ 16MHz [1] Idle mode @ 16MHz [2] Power down mode [3]		See note 7		 32 6.5 100	 mA mA µA
R_{RST}	Internal reset pull-down resistor			50	150	kΩ
C_{IO}	Pin capacitance		f = 1MHz		10	pF

NOTES:

1. The operating supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f$ = 5ns; $V_{IL} = V_{SS}$ +0.5V; $V_{IH} = V_{CC}$ – 0.5V; XTAL2 not connected; $\overline{EA}$ = RST = Port 0 = V_{CC}.
2. The idle mode supply current is measured with all output pins disconnected; XTAL1 driven with $t_r = t_f$ = 5ns; $V_{IL} = V_{SS}$ +0.5V; $V_{IH} = V_{CC}$ – 0.5V; XTAL2 not connected; $\overline{EA}$ = Port 0 = V_{CC}; RST = V_{SS}.
3. The power-down current is measured with all output pins disconnected; XTAL2 not connected; $\overline{EA}$ = Port 0 = V_{CC}; RST = XTAL1 = V_{SS}.
4. Capacitive loading on Port 0 and Port 2 may cause spurious noise pulses to be superimposed on the LOW level output voltage of ALE, Port 1 and Port 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make a 1-to-0 transition during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE line may exceed 0.8V. In such cases it may be desireable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
5. Capacitive loading on Port 0 and Port 2 may cause the HIGH level output voltage on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
6. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per Port pin: 10mA
   Maximum I_{OL} per 8-bit port –
   Port 0: 26mA
   Ports 1, 2, and 3: 15mA
   Maximum total I_{OL} for all output pins: 71mA.
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
7. See Figures 11 through 14 for I_{CC} test conditions.

AC ELECTRICAL CHARACTERISTICS[1, 2]

SYMBOL	FIGURE	PARAMETER	16MHz CLOCK		VARIABLE CLOCK		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	4	Oscillator frequency			1.2	16	MHz
t_{LHLL}	4	ALE pulse width	85		$2t_{CLCL}-40$		ns
t_{AVLL}	4	Address valid to ALE low	8		$t_{CLCL}-55$		ns
t_{LLAX}	4	Address hold after ALE low	28		$t_{CLCL}-35$		ns
t_{LLIV}	4	ALE low to valid instruction in		150		$4t_{CLCL}-100$	ns
t_{LLPL}	4	ALE low to $\overline{PSEN}$ low	23		$t_{CLCL}-40$		ns
t_{PLPH}	4	$\overline{PSEN}$ pulse width	143		$3t_{CLCL}-45$		ns
t_{PLIV}	4	$\overline{PSEN}$ low to valid instruction in		83		$3t_{CLCL}-105$	ns
t_{PXIX}	4	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	4	Input instruction float after $\overline{PSEN}$		38		$t_{CLCL}-25$	ns
t_{AVIV}	4	Address to valid instruction in		208		$5t_{CLCL}-105$	ns
t_{PLAZ}	4	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	5, 6	$\overline{RD}$ pulse width	275		$6t_{CLCL}-100$		ns
t_{WLWH}	5, 6	$\overline{WR}$ pulse width	275		$6t_{CLCL}-100$		ns
t_{RLDV}	5, 6	$\overline{RD}$ low to valid data in		148		$5t_{CLCL}-165$	ns
t_{RHDX}	5, 6	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	5, 6	Data float after $\overline{RD}$		55		$2t_{CLCL}-70$	ns
t_{LLDV}	5, 6	ALE low to valid data in		350		$8t_{CLCL}-150$	ns
t_{AVDV}	5, 6	Address to valid data in		398		$9t_{CLCL}-165$	ns
t_{LLWL}	5, 6	ALE low to $\overline{RD}$ or $\overline{WR}$ low	138	238	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AW}	5, 6	Address to $\overline{RD}$ or $\overline{WR}$	120		$4t_{CLCL}-130$		ns
t_{QW}	5, 6	Data setup time before $\overline{WR}$	288		$7t_{CLCL}-150$		ns
t_{QVWX}	5, 6	Data valid to $\overline{WR}$ transition	3		$t_{CLCL}-60$		ns
t_{WHQX}	5, 6	Data hold after $\overline{WR}$	13		$t_{CLCL}-50$		ns
t_{RLAZ}	5, 6	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	5, 6	$\overline{RD}$ or $\overline{WR}$ high to ALE high	23	103	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	8	High time	20		20		ns
t_{CLCX}	8	Low time	20		20		ns
t_{CLCH}	8	Rise time		20		20	ns
t_{CHCL}	8	Fall time		20		20	ns
Erase/write timer constant[3]							
$t_{E/W}$		Erase/write cycle time	10	40	10	40	ms
t_E		Erase time	5	40	5	40	ms
t_W		Write time	5	40	5	40	ms
t_S		Data retention time[4]	10		10		years
NE/W		Erase/write cycles[5]	50,000		50,000		cycles

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. The power-off fall-time of V_{CC} must be less than 1ms to prevent an overwrite pulse from being generated in the EEPROM which can cause spurious parasitic writing to EEPROM cells. If the V_{CC} power-off full-time is greater than 1ms, a power-off reset signal should be generated to prevent this condition from occurring.
4. Test condition: T_{amb} = +55°C.
5. Number of erase/write cycles for each EEPROM byte.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{\text{PSEN}}$
Q – Output data
R – $\overline{\text{RD}}$ signal
t – Time
V – Valid
W – $\overline{\text{WR}}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{\text{PSEN}}$ low.

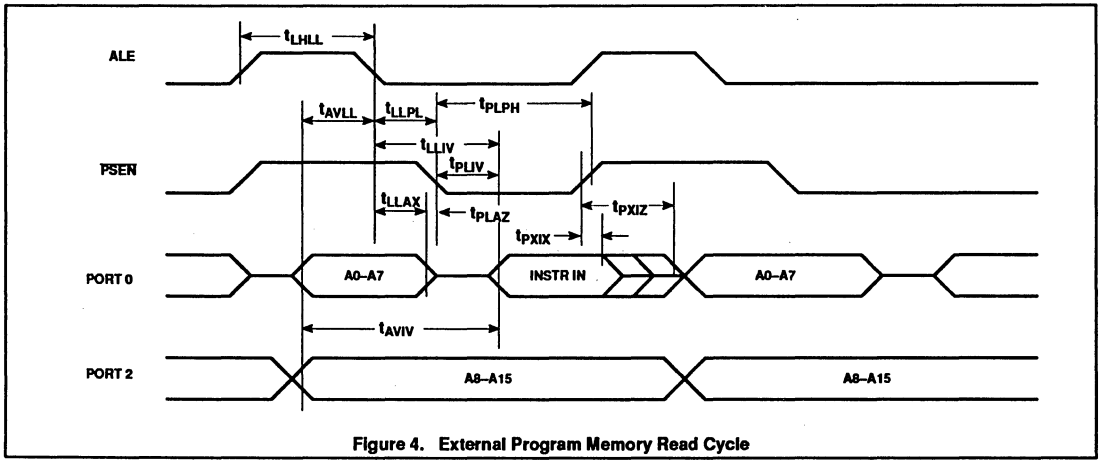


Figure 4. External Program Memory Read Cycle

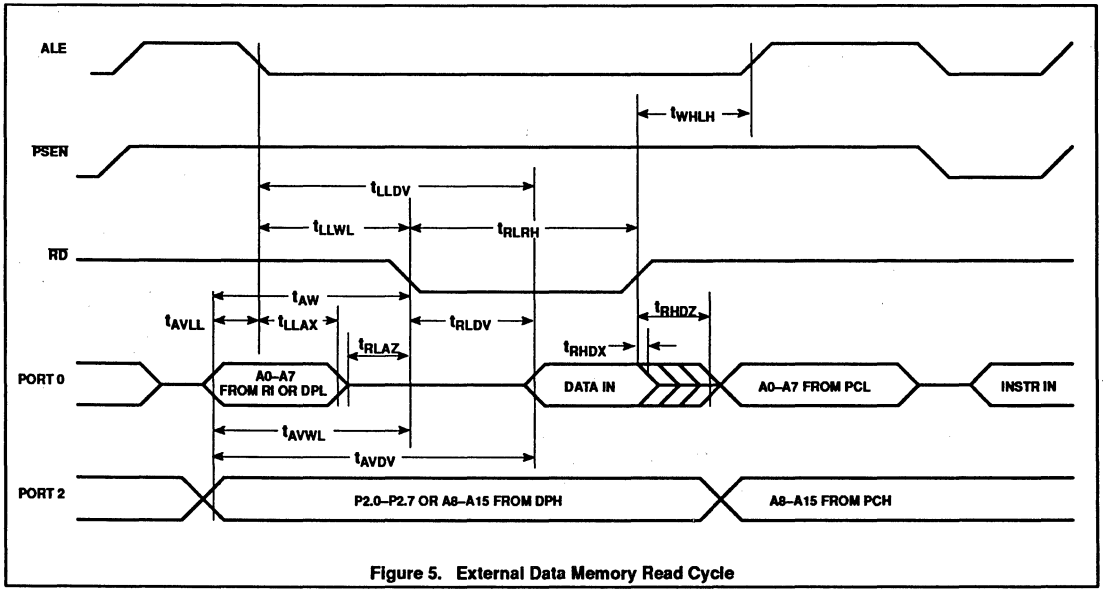


Figure 5. External Data Memory Read Cycle

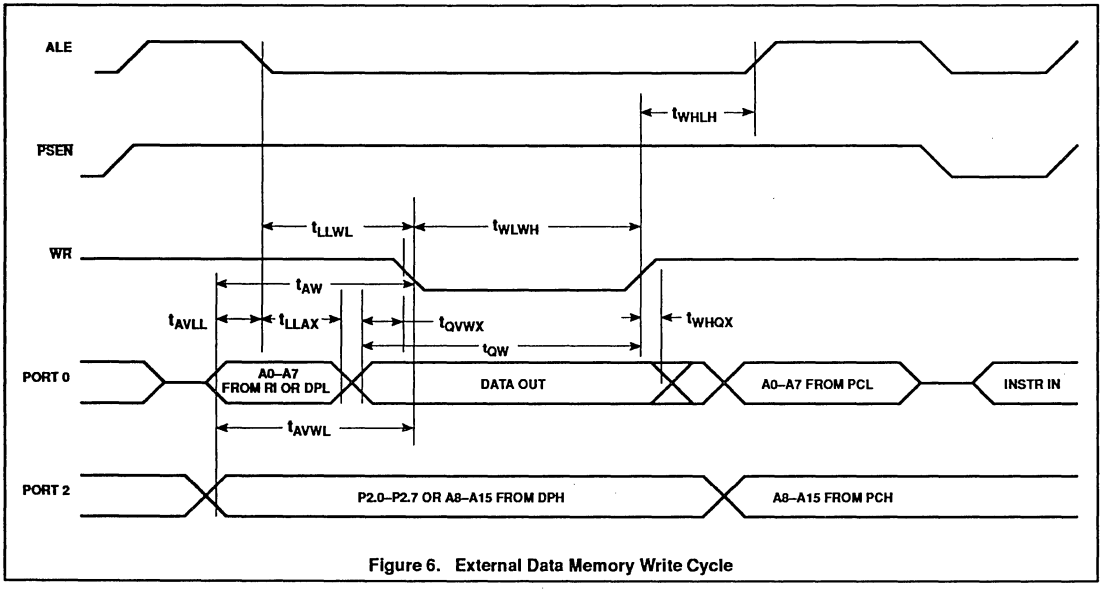


Figure 6. External Data Memory Write Cycle

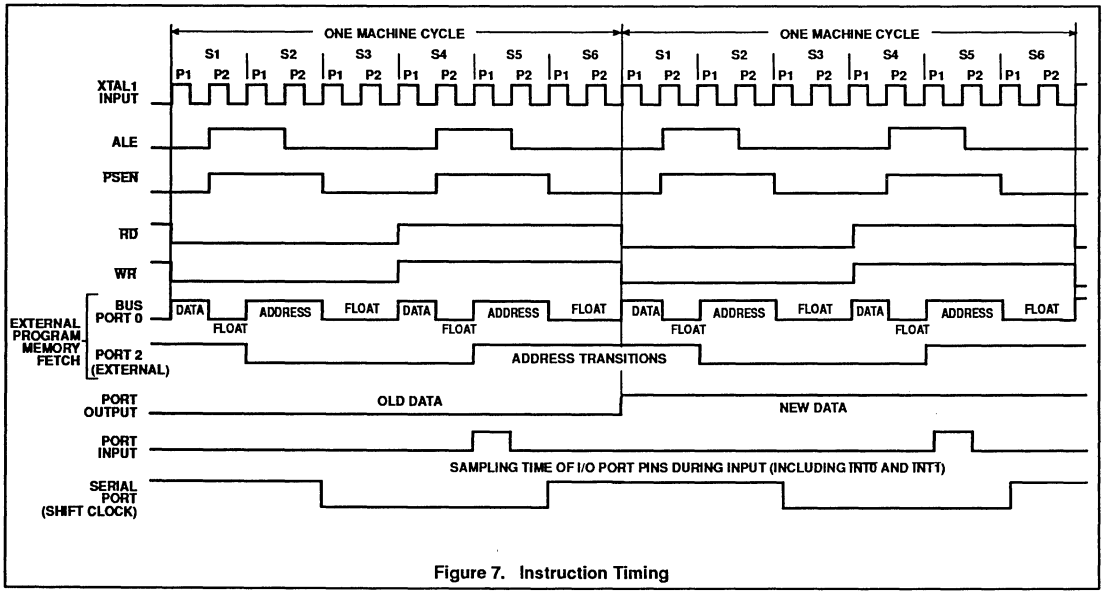


Figure 7. Instruction Timing

Table 3. External Clock Drive XTAL1

Oscillator circuitry: The capacities connected to the crystal should be: C1 = C2 = tbf.

SYMBOL	PARAMETER	VARIABLE CLOCK f = 1.2 – 16MHz		UNIT
		MIN	MAX	
t_{CLCL}	Oscillator clock period	63	833	ns
t_{HIGH}	HIGH time	20	$t_{CLCL} - t_{LOW}$	ns
t_{LOW}	LOW time	20	$t_{CLCL} - t_{HIGH}$	ns
t_r	Rise time	–	20	ns
t_f	Fall time	–	20	ns
t_{CY}	Cycle time[1]	0.75	10	ns

NOTE:
1. $t_{CY} = 12\ t_{CLCL}$.

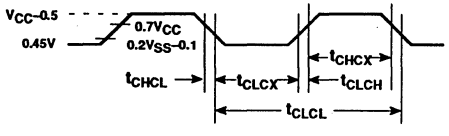
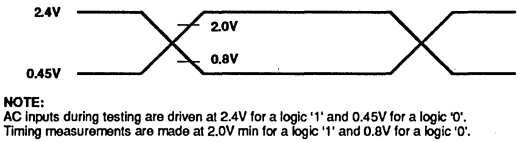
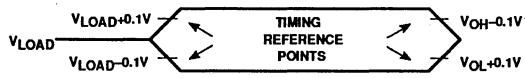


Figure 8. External Clock Drive



NOTE:
AC inputs during testing are driven at 2.4V for a logic '1' and 0.45V for a logic '0'. Timing measurements are made at 2.0V min for a logic '1' and 0.8V for a logic '0'.

Figure 9. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

Figure 10. Float Waveform

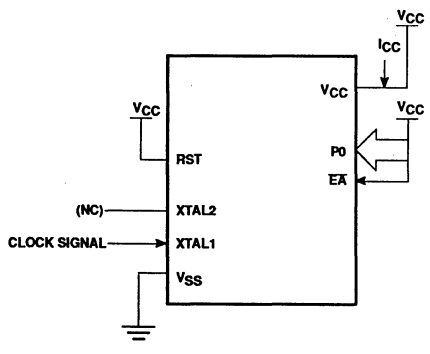


Figure 11. I_{CC} Test Condition, Active Mode
All other pins are disconnected

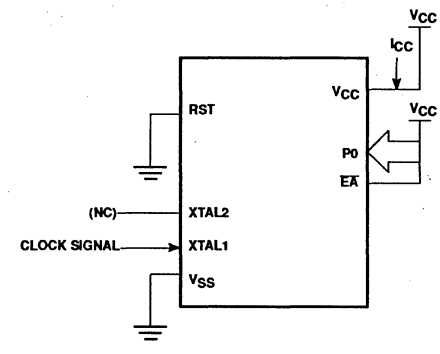


Figure 12. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

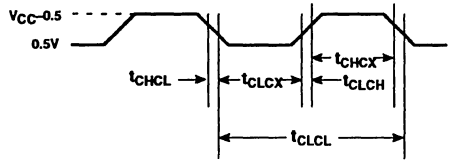


Figure 13. Clock Signal Waveform for I_{CC} Tests In Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

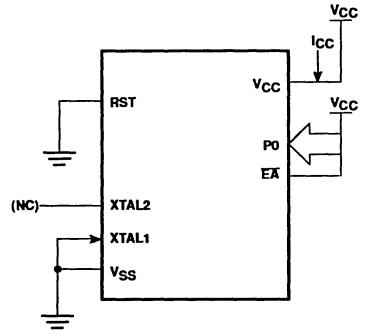


Figure 14. I_{CC} Test Condition, Power Down Mode
All other pins are disconnected.
V_{CC} = 2V to 5.5V

Secured 8-bit microcontroller — 83C852

FEATURES

- 8-bit CPU
- 6K bytes of user program memory (ROM), no external extension
- 256 bytes of RAM data memory (RAM), no external extension
- 2K bytes EEPROM:
  - EEPROM stores data or program
  - on-chip voltage multiplier for EEPROM ERASE/WRITE
  - ERASE/WRITE cycle time independent of the clock frequency
  - 10000 ERASE/WRITE cycles per bytes
  - 10 years non-volatile data retention
  - infinite number of READ cycles
  - error code correction
- Calculation unit for cryptographic calculations
- Security features
- Power-ON/OFF reset circuit
- Low frequency detector
- Two 16-bit timers
- Clock frequency range 1 MHz to 6 MHz; 1 µs cycle time with 6 MHz clock frequency
- Two I/O lines; only one I/O line is used in half-duplex, according to the ISO standards for the Smart Card applications; full-duplex communication can be performed with both I/O lines
- 5 interrupt sources from: I/O lines; Timer 0; Timer 1; EEPROM; Calculation unit
- Power-down and idle mode
- Two operating modes: test mode and user mode
- Single 5 volts power supply
- 6 pins: V_{DD}, V_{SS}, I/O1, I/O2, $\overline{RESET}$, CLK

GENERAL DESCRIPTION

The 83C852 single chip secured microcontroller is manufactured in an advanced 1.2 µCMOS process. It is a derivative of the 80C51 microcontroller family and has the same instruction set as the 80C51. It has been specially designed for conditional access in secure Smart Card applications and is implemented with the highest levels of security.

Its internal calculation unit speeds-up cryptographic calculations using Public Key Algorithms.

Cryptographic calculations
At f_{CLK} = 6 MHz: X^e mod.n is performed in 1.5 s typical, with 512 bit operands.

External communications can be performed through a serial interface (I/O) according to ISO standards. The serial interface must be controlled by application software for access to the 83C852 internal memory.

The 83C852 contains a 6K bytes READ only memory (user ROM); a 256 bytes READ/WRITE data memory (RAM); 2K bytes electrically erasable programmable READ only memory (EEPROM); two I/O lines; two 16-bit timers; five vectorized interrupt sources; 33 Special Function Registers (SFRs) and a Calculation Unit to speed-up the execution time of public keys and secret keys cryptographic algorithms.

The 83C852 operates with a single 5 volts power supply and at a maximum clock frequency of 6 MHz. The instruction set consists of over 100 instructions: 49 one-byte, 45 two-byte and 17 three-byte. With an input clock frequency of 6 MHz, 58% of the instructions are executed in 1 µs and 40% in 2 µs.

QUICK REFERENCE DATA

SYMBOL	PARAMETER	CONDITION	MIN.	MAX.	UNIT
V_{DD}	supply voltage range		4.5	5.5	V
I_{DD}	supply current	f_{CLK} = 3.57 MHz	–	10	mA
I_{DD}	supply current: operating modes	f_{CLK} = 6.0 MHz	–	15	mA
I_{ID}	supply current: idle mode	f_{CLK} = 6.0 MHz	–	3	mA
P_{tot}	total power dissipation		–	1	W
T_{stg}	storage temperature range		–65	150	°C
T_{amb}	operating ambient temperature range		0	70	°C

ORDERING INFORMATION

EXTENDED TYPE NUMBER	PACKAGE			
	PADS	PAD POSITION	MATERIAL	CODE
83C852 die	6	X Y coordinates	die	–
83C852P	tbf	tbf	plastic DIL	tbf

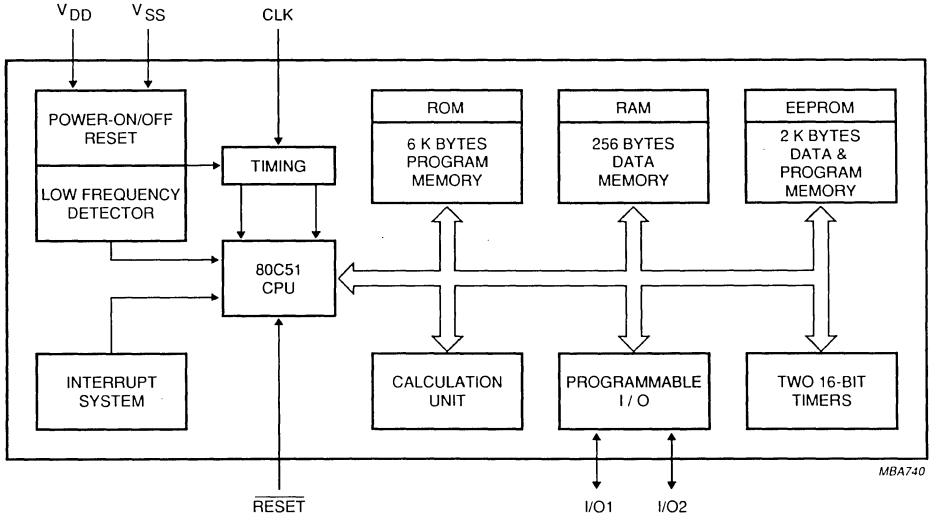


Fig.1 Block diagram.

PINNING

SYMBOL	PIN (PAD)	DESCRIPTION
V_{DD}	C1	+5 volts power supply pin during normal operation, idle mode and power-down mode
$\overline{RESET}$	C2	active LOW input that initializes the processor
CLK	C3	external clock input. The internal clock frequency = the external clock frequency
V_{SS}	C5	ground
I/O1	C7	quasi bi-directional port (TTL compatible), the user's program must include routines able to handle an asynchronous serial communication through a single I/O port (for half-duplex)
I/O2	C8	quasi bi-directional port (TTL compatible)

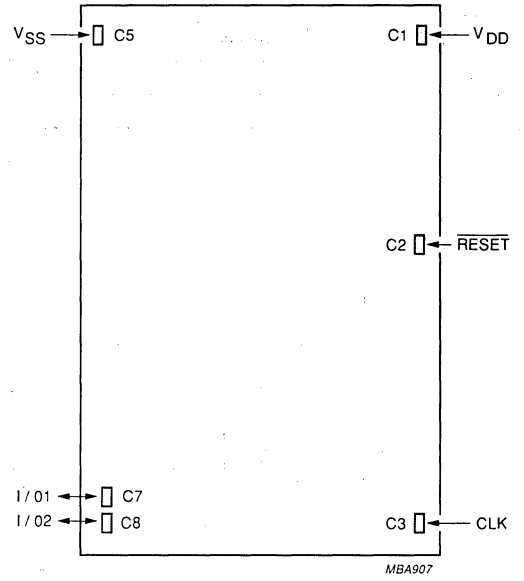
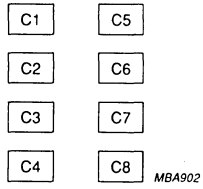


Fig.2 Pinning diagram.

ASSIGNMENT OF ISO/83C852 SMART CARD CONTACTS



CONTACTS	ISO ASSIGNMENTS	83C852 ASSIGNMENTS
C1	V_{CC}	V_{DD}
C2	$\overline{RESET}$	$\overline{RESET}$
C3	CLK	CLK
C4	reserved	not connected
C5	GND	V_{SS}
C6	V_{PP}	not connected
C7	I/O	used for I/O1
C8	reserved	reserved for I/O2

contact assignments are specified in part 2 of ISO 7816

Fig.3 Contact assignments.

FUNCTIONAL DESCRIPTION

General

The 83C852 is specially designed for secured applications such as conditional access and transactions in a Smart Card environment. It has a Calculation Unit which makes the microcontroller dedicated to asymmetric crypto systems. Special Function Registers (SFRs) are available to the user to manipulate memory transactions and calculation unit operations.

The address bus of the EEPROM is mixed to prevent fraudulent access and optical scanning. The EEPROM has an hardware error code correction which guarantees the data content integrity.The 83C852 is able to read and modify a part of the internal program memory contained in EEPROM.

The 83C852 has two software selectable modes of reduced activity for further power reduction, the idle mode and power-down mode:

- the idle mode freezes the CPU while allowing the RAM, the timers and the interrupt system to continue functioning.
- the power-down mode saves the RAM content and disables all other chip functions.

The 83C852 has 33 SFRs available for use by the user (see Table 23). The functional descriptions and usage of the SFRs as they co-relate to the RAM, EEPROM and the calculation unit activities, are described within the following sections.

Memory organisation

The central processing unit (CPU) manipulates operands in three memory spaces, (see Fig. 4) these are:

- 6K-byte internal program memory (ROM)
- 2K-byte program and data memory (EEPROM)
- 256-byte internal data memory (RAM).

The 256-byte internal RAM memory address space is sub-divided into:

- 128-byte internal data RAM locations 00 to 7FH. This address space is accessible with direct and indirect addressing
- 128-byte internal data RAM locations 80H to FFH. This address space is accessible with indirect addressing only
- 128-byte Special Function Register (SFR) address space 80H to FFH. This address space is parallel to the upper 128 byte RAM. It is accessible with direct addressing only. 33 SFRs reside inside this area, the remaining address space in between is unused.

EEPROM

The EEPROM has a capacity of 2K bytes (words) see Fig. 5. With its built-in error correction hardware the EEPROM is a very reliable non-volatile memory. In addition to each single stored data byte, 4 extra "parity" bits are stored in EEPROM. Single-bit errors per byte are automatically corrected when reading the memory. It can be accessed by both CPU and calculation unit (however, not at the same time).

Programming of the EEPROM is completely controlled by the EEPROM's sequencer. The EEPROM can be used either both as data memory and program memory for the CPU, or as data memory for the calculation unit.

EEPROM AS DATA MEMORY:

When the CPU executes opcodes from internal ROM (program address < 8000H), the EEPROM can be used as a data memory. The communication between CPU and EEPROM is performed via 6 SFRs (see Table 1), these comprise:

- 2 EEPROM (SFRs) address registers (HIGH and LOW address byte)
- 1 EEPROM (SFR) data register for READ and WRITE operations
- 2 EEPROM (SFRs) control registers to select the various operating and test modes
- 1 EEPROM (SFR) timer register to adapt the ERASE/WRITE time to the operating clock frequency.

EEPROM AS PROGRAM MEMORY:

When the program counter is higher than 7FFFH, the EEPROM is used as a program memory. The CPU fetches opcodes directly from the EEPROM. EADRH, EADRL1 and EDAT registers cannot be written. Their contents in this mode are irrelevant.

Reading data from the EEPROM can still be done with the MOVC instruction, but EEPROM write operation is not possible. The EEPROM can only be written by software executed from the ROM area (program address < 8000H).

EEPROM AS DATA MEMORY FOR THE CALCULATION UNIT:

Communication between calculation unit and EEPROM is performed via SFRs (see Table 1), these comprise:

- 2 EEPROM SFRs address registers (one HIGH and one of two alternate LOW address bytes)
- calculation unit SFRs.

The EEPROM data output is directly connected via a special bus to the data registers of the calculation unit. The calculation unit has direct READ access to the EEPROM.

During memory access, the EEPROM is addressed by one of two address pointers EADRL1 or EADRL2. Both pointers are loaded by the CPU and decremented by the calculation unit's sequencer. EADRL1 or EADRL2 supply the LOW byte (LSB) of the EEPROM address. The HIGH byte (MSB) of the EEPROM address is taken directly from the EADRH register. The access to the EEPROM from the calculation unit, is controlled by the calculation unit SFRs.

When the access to EEPROM by the calculation unit is active, the EEPROM is not accessible by the CPU, neither as data memory nor as program memory. When the calculation unit is operating, but not accessing the EEPROM, the CPU can READ, WRITE and EXECUTE EEPROM.

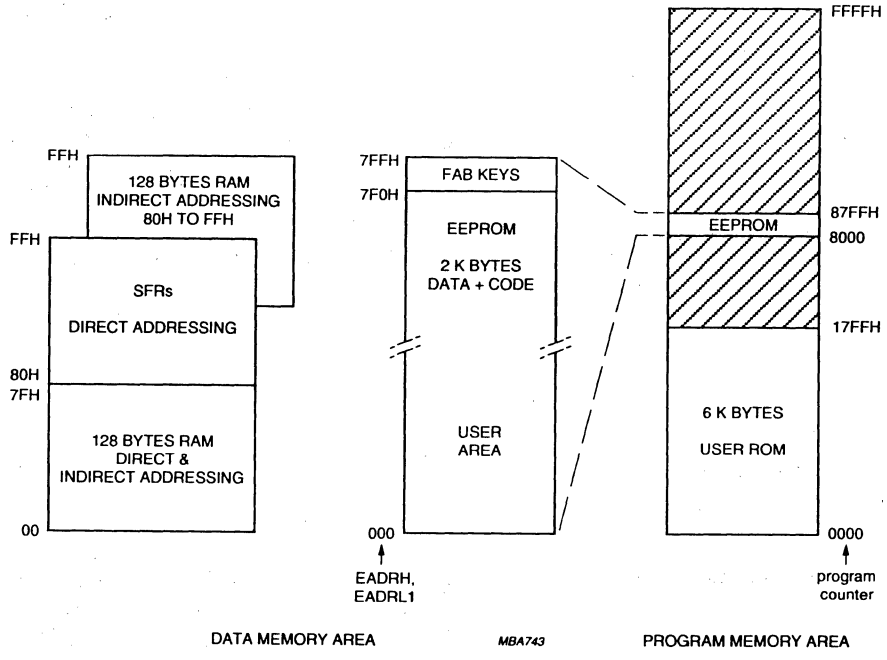


Fig.4 Memory map.

EEPROM SFRs

Table 1 provides a listing of the EEPROM associated SFRs:

Table 1 EEPROM SFRs

NAME	SFR ADDRESS	FUNCTION
The communication between the CPU and the EEPROM is performed via the following SFRs:		
EADRL1	0F2H	address register (LSB)
EADRH	0F3H	address register (MSB)
EDAT	0F4H	data register
ETIM	0F5H	EEPROM timer register
ECNTRL1	0F6H	control register for normal operation modes
ECNTRL2	0F7H	control register for special test modes
The communication between the calculation unit and the EEPROM is performed via the following SFRs:		
EADRL2	0F1H	address pointer 2: LSB of the EEPROM address (WRITE: reload EADRL2 register; READ: read counter)
EADRL1	0F2H	address pointer 1: LSB of the EEPROM address (WRITE: reload EADRL1 register; READ: read counter)
EADRH	0F3H	address register (MSB)
Two address pointers (SFRs) are necessary for EEPROM access by the calculation unit. SFRs EADRH and either EADRL1 or EADRL2		

EEPROM SFR descriptions

EADRH

EADRH is an 8-bit register (SFR), used as an address pointer, it is loaded from the CPU and contains the highest byte (MSB) of the EEPROM address. Only bits EADRH.7, 2, 1, 0 are relevant. Default value after reset is '10000000B'.

EADRL1, EADRL2

EADRL1 is an 8-bit register (SFR), used as an address pointer, it is loaded from the CPU. Its contents are transferred into an 8-bit down counter which provides the LOW byte (LSB) of the EEPROM address. The CPU has WRITE access to the EADRL1 register and READ access to the down counter.

Default value after reset of both the EADRL1 register and associated down counter are 00H. The transfer of EADRL1 to the down counter and the decrement are controlled by the calculation unit. The behaviour of EADRL1 depends on whether or not there is an access in progress from the calculation unit to the EEPROM.

No access from the calculation unit to the EEPROM:

- when there is no active access to the EEPROM from the calculation unit, the contents of the EADRL1 register are continuously loaded into its associated down counter. There is no hardware-decrement of the down counter. The combination of EADRL1 plus it's down counter behaves therefore as a normal register. In this mode, the LOW address byte of the EEPROM is always supplied by EADRL1. EADRL2 behaves similar to EADRL1 but it is not used for EEPROM addressing. Its contents are irrelevant.

Calculation unit access to the EEPROM:

- EADRL1 is the EEPROM LOW address pointer used to address an operand (Ai) stored in the EEPROM. At the beginning of the calculation unit's computation cycle, the address content of the EADRL1 is loaded into it's associated down counter. During the calculation, further EADRL1 address transfers to the down counter stop, whilst the down counter is decremented by the calculation unit's sequencer. During a calculation, the EADRL1 register can be reloaded from the CPU with a new address. This new address will then be used during the next calculation.

- EADRL2 (SFR) is the second EEPROM LOW address pointer which is required for some operations of the calculation unit. It is used to address an operand (Xi) stored in the EEPROM. The function of EADRL2 is similar to EADRL1 function. EADRL2 cannot be used as a second address register for normal CPU access to the EEPROM.

Data register EDAT

This register (SFR) is used to read data from the currently addressed EEPROM byte. When EDAT is written during BYTE MODE, it's contents will be programmed into the addressed EEPROM byte. When EDAT is written during ROW ERASE or BLOCK ERASE mode, the ROW ERASE or BLOCK ERASE operation is started. In this mode, the data written to EDAT is then irrelevant. The ECNTRL1 status bits EWP and IFE indicate whether the EEPROM ERASE/WRITE operation is still active. Whilst the EEPROM programming is in progress, rewriting data to EDAT is not allowed.

Timer register ETIM

The ETIM timer register (SFR) is required to adapt the ERASE/WRITE time to the clock frequency. ERASE (t_e) and WRITE (t_w) times of 5 ms each are required. The user has to ensure that the ERASE or WRITE time is neither too short nor too long. Table 2 gives values for ETIM register for given clock frequencies. ETIM has to be loaded by software in advance to the first ERASE/WRITE operation. ETIM's default value after reset is '08H'.

Table 2 ETIM timer values
The general formula is: Value (decimal) =(f_{CLK} kHz/102.4)–2

OPERATING f_{CLK} MHz	VALUES FOR ETIM			
	BINARY		HEXADECIMAL	DECIMAL
	MSB	LSB		
1.0	0000	1000	8	8
2.0	0001	0010	12	18
3.0	0001	1011	1B	27
3.57	0010	0001	21	33
4.0	0010	0101	25	37
4.92	0010	1110	2E	46
5.0	0010	1111	2F	47
6.0	0011	1001	39	57

Control register ECNTRL1

ECNTRL1 is the control register (SFR) for the several user operation modes of the EEPROM.

Table 3 ECNTRL1 SFR

IFE	EEINT	EWP	–	OPERATION MODE SELECT			
7	6	5	4	3	2	1	0
			BYTE MODE →	0	0	0	0
			ROW ERASE →	1	1	0	0
			BLOCK ERASE →	1	0	1	0
			TEST MODE →	1	1	1	1

Table 4 Description of the ECNTRL1 bits

SYMBOL / PARAMETER	FUNCTION
ECNTRL1.7	
IFE, interrupt flag EEPROM	set by the EEPROM sequencer after completion of an EEPROM WRITE access, or set and reset by software. When (IFE is set 1 and EEINT is set 1), an interrupt request is done. Interrupt vector 0023H will be forced if the bits EA and EE inside the interrupt Enable Register IE are also set 1.
ECNTRL1.6	
EEINT, enable EEPROM interrupt	set and reset by software. Enables an EEPROM interrupt request when HIGH.
ECNTRL1.5	
EWP, ERASE/WRITE in progress	set and reset by the EEPROM sequencer. EWP is active HIGH during EEPROM write operations. Consecutive write operations to EDAT are not allowed as long as EWP is set. EWP cannot be set or reset by software.
ECNTRL1.4	
–	reserved.
ECNTRL1 bits 3, 2, 1, 0	
operation mode select: BYTE MODE (0000)	normal E^2PROM mode, default mode after reset. In this mode READ or WRITE access to one byte at a time is possible.
READ mode	This is the default mode when BYTE MODE is selected. The contents of the addressed byte are available in the data register EDAT.
WRITE mode	This mode is activated after loading of the data register EDAT with the data byte to be written. Before writing EDAT, the address registers EADRL1 and EADRH must be loaded first. Depending on the previous contents of the addressed memory cells, the EEPROM sequencer decides whether to do a WRITE cycle (t_w), or a combined ERASE/WRITE cycle (t_e + t_w). A WRITE cycle is carried out when the previous memory content has been 00H. Otherwise an ERASE/WRITE cycle is carried out.

SYMBOL / PARAMETER	FUNCTION
operation mode select: ROW ERASE (1100)	in this mode the contents of the addressed memory row will be erased. The three LSB's of EADRL1 are not significant, i.e. 8 bytes addressed by EADRL1 will be cleared in the same time normally needed to clear one single byte ($t_{ROW\ ERASE} = t_e$). The ROW ERASE operation can be started by writing the EDAT register. The data that is written to EDAT is not significant. Writing of the erased 8 memory cells then takes only 8 WRITE cycles. This is much faster than writing 8 data bytes without a previous ROW ERASE. Such an operation would have taken a total = $8\ t_e + 8\ t_w$.
operation mode select: BLOCK ERASE (1010)	in this mode all memory cells of the EEPROM will be cleared. The BLOCK ERASE operation can be started by writing EDAT. The contents of the data and address registers EDAT, EADRL1, EADRH are don't care.
operation mode select: TEST MODE (1111)	the selection of a specific EEPROM TEST MODE within the ECNTRL2 register is only possible when the ECNTRL1 register is switched in advance to TEST MODE.

Control register ECNTRL2

ECNTRL2 is the control register (SFR) for the several test modes of the EEPROM.

Table 5 ECNTRL2 SFR

READ ONLY				READ/WRITE			
SP3	**SP2**	**SP1**	**SP0**	**TM3**	**TM2**	**TM1**	**TM0**
7	6	5	4	3	2	1	0
NO TEST →				0	0	0	0
READ EEPROM WITHOUT ERRORCORRECTION →				1	0	1	1

Table 6 Description of the ECNTRL2 bits

SYMBOL / PARAMETER	FUNCTION
ECNTRL2 bits 7, 6, 5, 4	
SP3, SP2, SP1, SP0	this part of the ECNTRL2 register is READ only. The upper 4 bits of ECNTRL2 carry either the syndrome word which is generated by the EEPROM error correction logic or the parity bits stored in parallel to the data word in EEPROM memory. The syndrome word is always output during BYTE MODE READ (EWP = 0). A value of '0000B' means that no error has been detected/corrected. The parity bits are output during READ EEPROM WITHOUT ERRORCORRECTION TESTMODE or while EWP = 1.
ECNTRL2 bits 3, 2, 1, 0	
TM3, TM2, TM1, TM0	this part of ECNTRL2 register is READ/WRITE. The lower 4 bits of ECNTRL2 are used to select one of the EEPROM test-modes. The selection of any test-mode is only possible if ECNTRL1 has been set to 'XXXX1111B' before. Otherwise TM3, TM2, TM1, TM0 are held at '0000B'.

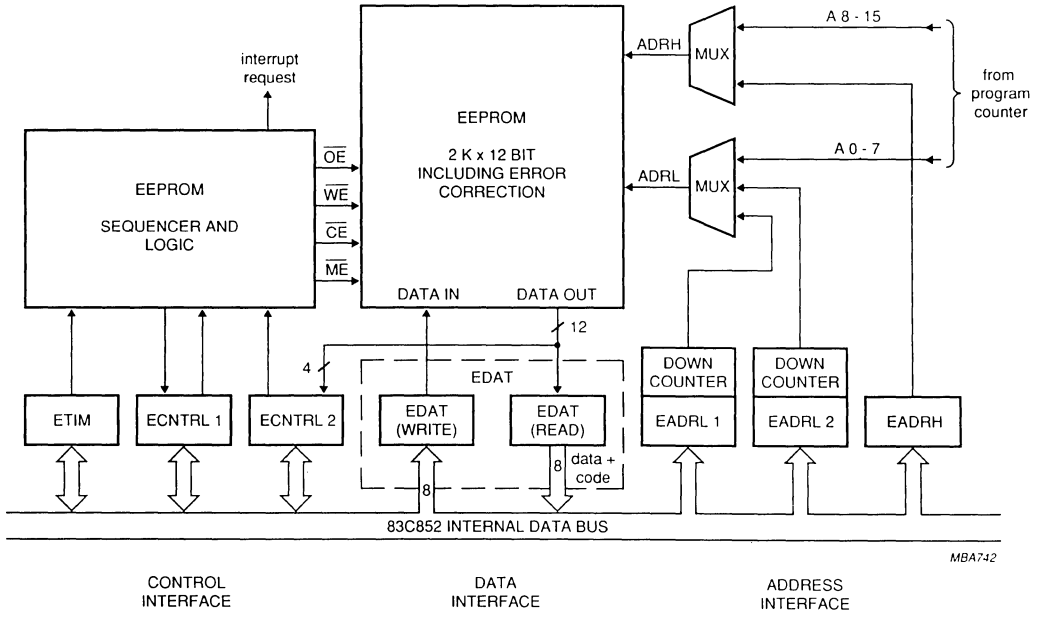


Fig.5 EEPROM Interface.

RAM

The RAM has a capacity of 256 bytes. It can be accessed by both CPU and calculation unit (however, not at the same time). The CPU has full READ and WRITE access to the RAM only when the calculation unit is not operating.

When the calculation unit is operating, it reads and writes data from/to the RAM via a separate channel. In order to avoid possible access conflicts, the CPU cannot read or write the RAM at this time. This condition places some restrictions to the operations of the CPU:

- no subroutine calls possible while calculation unit is active
- register operations (e.g. MOV A,R0) are not possible
- stack pointer operations (PUSH, POP) are not possible
- interrupt requests are not granted while the calculation unit is active.

In the RAM address space, only the SFRs can be read and written by the CPU whilst the calculation unit is active.

RAM address pointers

The calculation unit uses 4 RAM pointers to address the RAM during it's direct memory access: AIPR, XIPR, AOPR and APR.

ADDRESS POINTERS AIPR, XIPR, AOPR

AIPR and XIPR pointers address the operand fields Ai resp. Xi inside RAM, while AOPR addresses a RAM area Ao where the calculation result is to be stored. Each pointer consists of an 8-bit register with associated 8-bit down counter. The counter provides a RAM address. It is parallel loaded from it's register.
The CPU has WRITE access to the registers and READ access to the counters. Default values after reset for all registers and counters are 00H. The data transfer from the registers to their down counters and the counter decrement are controlled by the calculation unit.

ADDRESS POINTER APR

This is an 8-bit up counter which is used to address the A[3-0] operand field in RAM. It can be read and written by the CPU. Default value after reset is 00H. APR is incremented under the control of the calculation unit.

RAM address pointer modes

The following RAM address pointer modes apply:

The calculation unit is at standby:

- there is no direct memory access from the calculation unit
- the down counters are continuously loaded from the AIPR, XIPR and AOPR registers
- because RAM is not addressed by any of the RAM pointers, their contents are irrelevant at this time. In advance to a calculation, the CPU has to load the RAM pointers with the start addresses of the operand and result fields. AIPR, XIPR and AOPR have to be loaded with the LSB's address, while APR has to be loaded with the MSB's address of the data field.

The calculation unit is active:

- there is direct RAM memory access from the calculation unit
- the data transfer from the registers to their associated down counters is stopped
- the address pointers AIPR, XIPR and APR address the operands Ai, Xi and A[3-0] while AOPR addresses the Ao result area in RAM
- while the up and down counters are incremented/ decremented under control of the calculation unit, the CPU can reload the registers with new addresses, ready for transfer to their down counters at the beginning of the next calculation cycle.

CALCULATION UNIT

This unit computes, with it's associated software, any exponential functions like x^e mod.n. It has been designed to optimize the calculation time of exponent modulo N. It uses 196 bytes of RAM for 512-bit length operands.

CALCULATION UNIT PERFORMANCE

At f_{CLK} = 6 MHz: X^e modulo N is performed in 1.5 s typical, with 512 bit operands.

To reach this speed, the calculation unit's architecture provides:

- fast multiplication and addition
- fast carry handling
- fast data transfers to fetch operands from RAM or EEPROM and to store results in RAM
- simultaneous operation of both CPU and calculation unit.

The calculation unit does not carry out a complete exponentation in one step. However, it provides a set of basic instructions, from which the complete exponentation algorithm can be built by a dedicated software. All of these basic instructions operate on data-fields inside RAM and EEPROM. The width of these data-fields is variable. A typical operand width is 512 bits.

The basic operation of the calculation unit is to multiply either a 24-bit number or a 32-bit number with a long-word (e.g. 512-bit) and adding the result to another long-word. Further XOR and shift operations may be carried out to give the final result. With a 32-bit number this operation completes in typically 45 µs at 6 MHz clock frequency.

CALCULATION UNIT BASIC OPERATION

The basic operation of the calculation unit is:	
$B = [(A + a * X) \oplus value] * 2^n$	Where "A","B" and "X" are large numbers, "a" is a 3 or 4 byte part of the large number "Y" and "value" is either one same byte used for each result byte or the large number "X". "n" is the number of bit-shifts for the calculation result, n can be either 0 or 32.
Operation set of the calculation unit:	
$B = (A + a * X) \oplus value * 2^0$	multiply and accumulate step (see Fig. 6).
$B = (0 + a * 0) \oplus value * 2^0$	memory initialization (see Fig. 7).
$A = (A + a * 0) \oplus value * 2^{32}$	4 byte shift (see Fig. 8).
$B = (A + a * 0) \oplus value * 2^0$	memory transfer (see Fig. 9).
$B = (A + a * X) \oplus value * 2^{32}$	multiply, accumulate step with shift (see Fig. 10).
Force A = 0 or X = 0 and shift depend on the contents of the registers CMD and CMDSTAT.	
A calculation example of D = M*C Mod N is shown in Fig.11, where D, M, C and N are large numbers of n-bit length.	

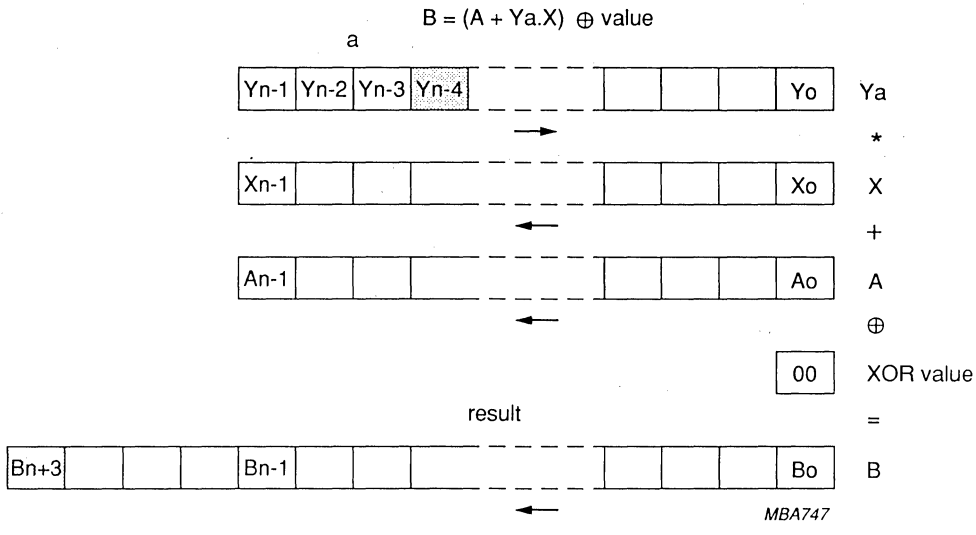


Fig.6 Multiply and accumulate step.

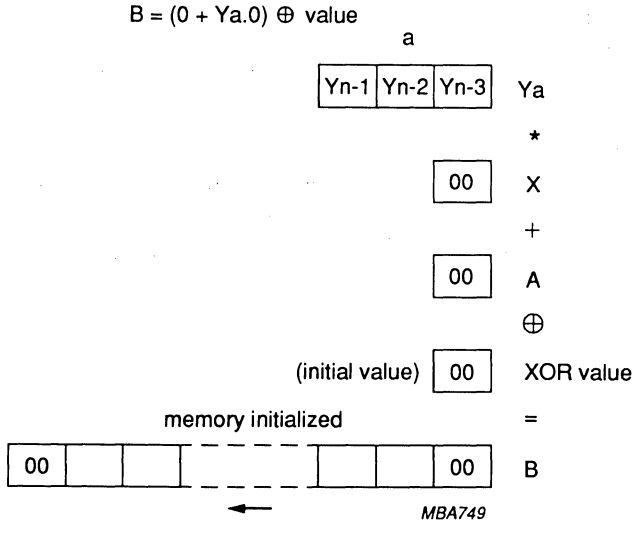


Fig.7 Memory initialization.

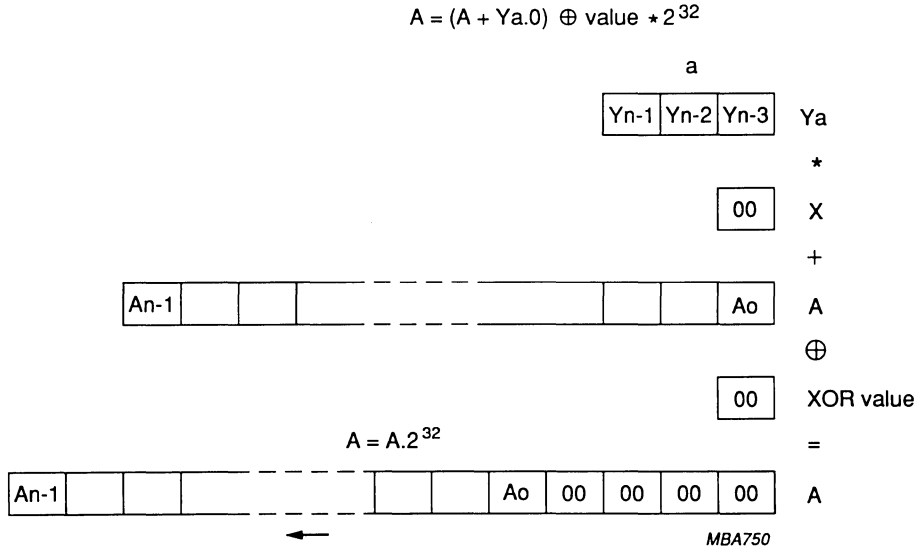


Fig.8 Shift (up to 4 bytes).

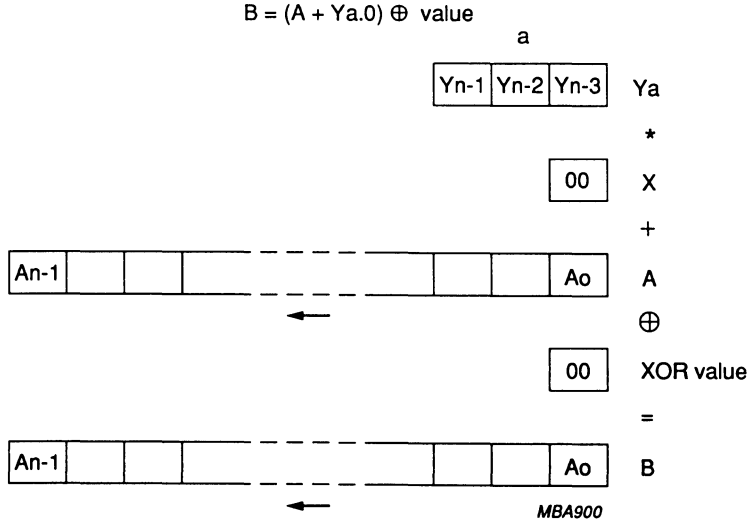


Fig.9 Memory transfer.

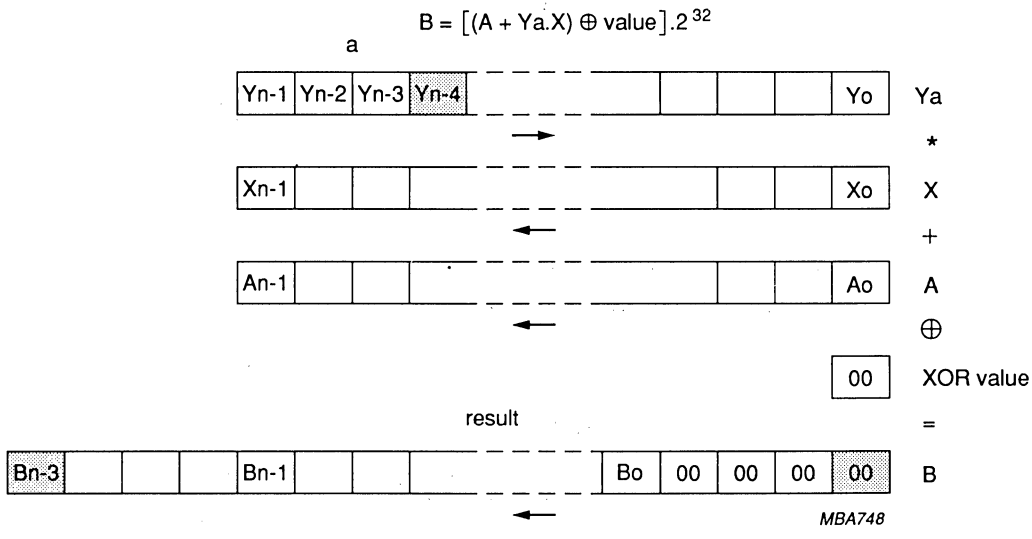


Fig.10 Multiply accumulate step and shift.

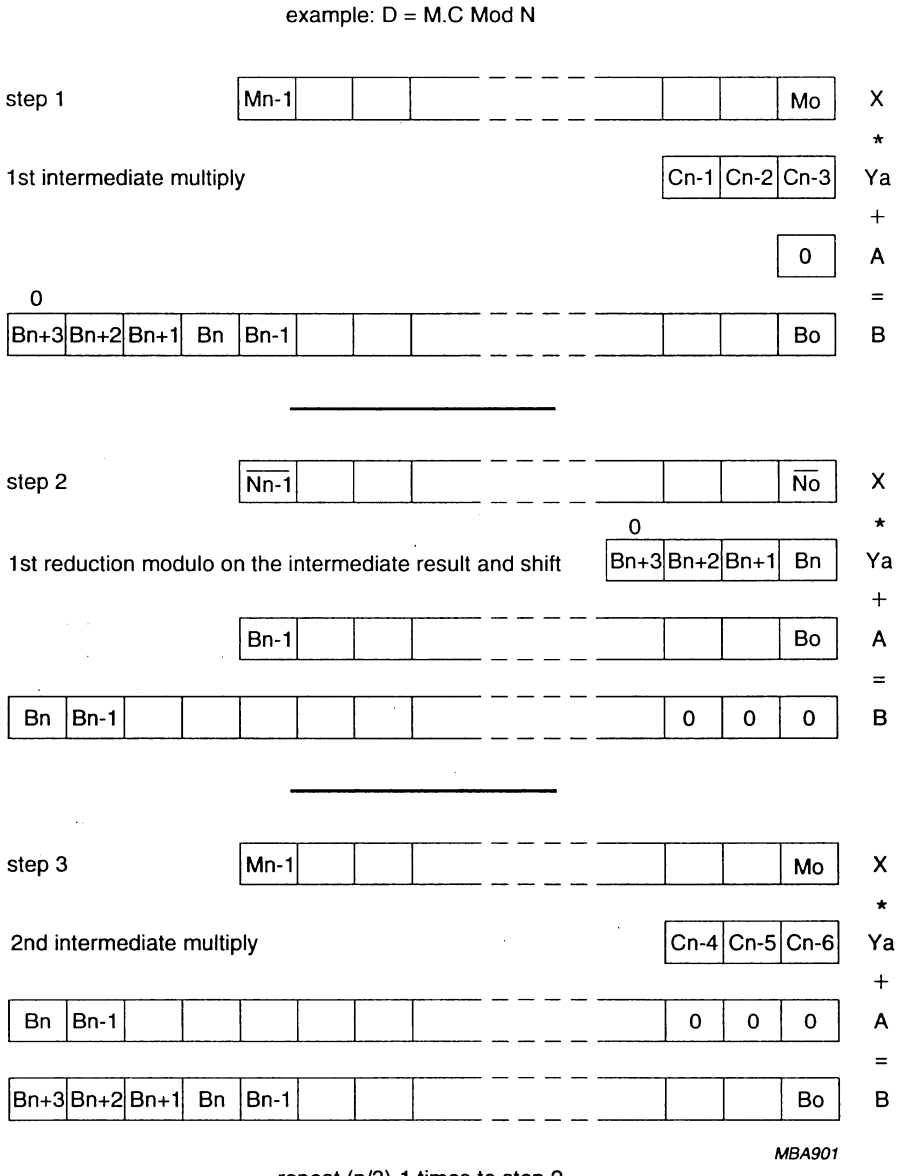


Fig.11 Calculation example.

Calculation unit related SFRs

12 Special Function Registers (SFRs) are related to the calculation unit, see Table 7 and Figure 12.

Memory access registers

The calculation unit has direct READ and WRITE memory access to the RAM and READ-only access to the EEPROM. The unit uses four 8-bit wide RAM pointers to address operands and result inside RAM and two 8-bit pointers to address operands inside EEPROM. The MSB of the EEPROM address is supplied by the EADRH register. 5 out of the 6 pointers are pipelined. This allows the CPU to initialize these registers while a calculation is busy.

Table 7 Calculation unit related SFRs

SYMBOL	ADDRESS	FUNCTION
AIPR	A4H	RAM address pointer for Ai input operand; note 1
EADRL1	F2H	EEPROM address pointer for Ai input operand; note 1
XIPR	A5H	RAM address pointer for Xi input operand; note 1
EADRL2	F1H	EEPROM address pointer for Xi input operand; note 1
APR	A6H	RAM address pointer for A[3-0] input operand
AOPR	A7H	RAM address pointer for Ao result output
CMD	99H	command register
CMDSTAT	98H	command and status register
CNTCYCL	F9H	cycle counter
CXOR	A3H	XOR operand register
WRLIM	FBH	WRITE operation limit register
RDLIM	FAH	limits register of the READ operands Xi and Ai.

Note

1. Ai operand and Xi operand can be stored in either the RAM or the EEPROM.

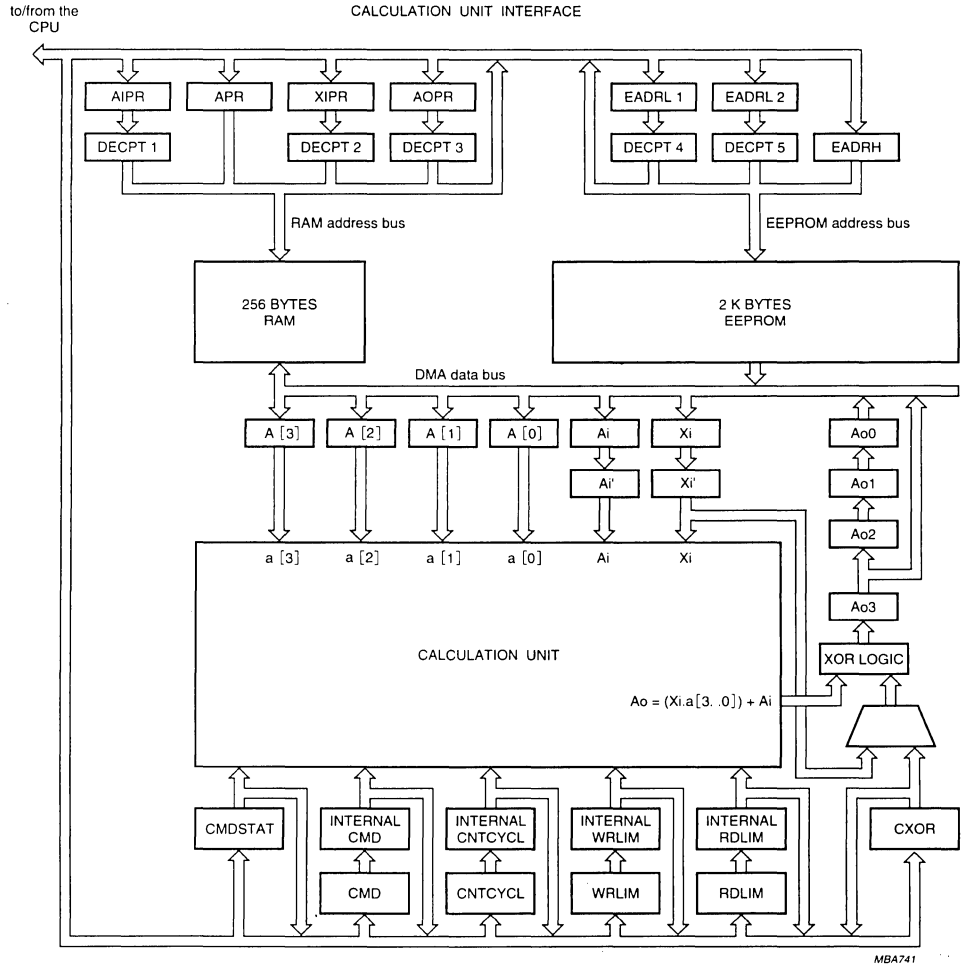


Fig.12 Calculation unit and memory access.

Command and status registers CMD and CMDSTAT

The calculation unit has two 8 bit registers (SFRs) for commands and status. SFR CMD is used for commands only, SFR CMDSTAT is used for both commands and status (2 command bits and 3 status bits).

7 bits of the CMD register are pipelined (see Table 8). At the beginning of a calculation, the contents of these seven bits are transferred into an internal command register that controls the calculation unit's sequencer.

This allows the CPU to re-initialize the CMD register for the next calculation while the current calculation is still busy.

The CMDSTAT SFR (see Table 10) is not pipelined and may not be reloaded by the CPU whilst the calculation unit is active. Both CMD and CMDSTAT registers are cleared at reset. Tables 9 and 11 describe the function of single status and control bits within the CMD and CMDSTAT registers.

Table 8 CMD SFR

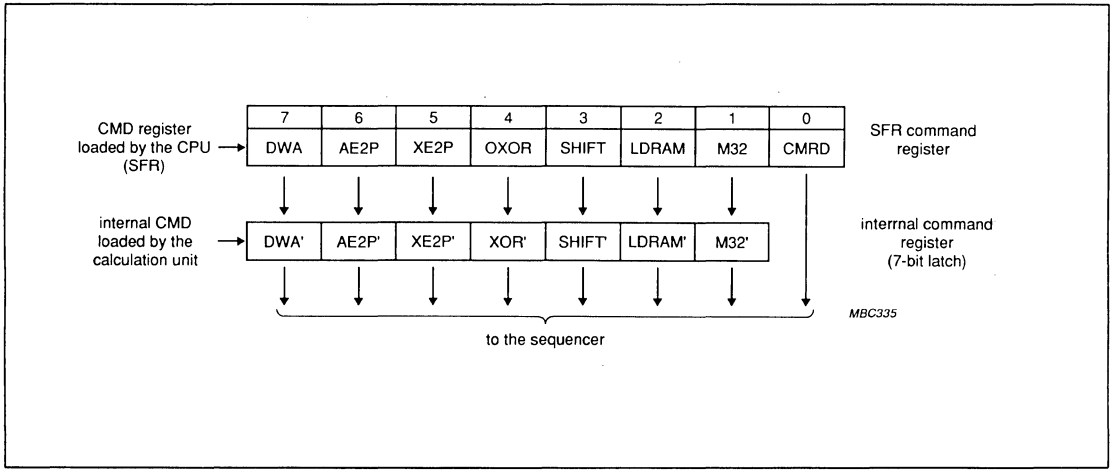


Table 9 Description of the CMD bits

SYMBOL / PARAMETER	FUNCTION
CMD.7	
DWA = disable output write	set and reset by the CPU. When the DWA bit is set 1, the writing operation of the Ao result into RAM is disabled. The AOPR address counter is not decremented. When DWA bit is reset 0, the Ao result bytes are written into RAM.
CMD.6	
AE2P = input Ai operand from the EEPROM	set and reset by the CPU. When the AE2P bit is set 1 the EEPROM is addressed by AIPE (EADRL1) address pointer and the Ai operand is read from the EEPROM. When AE2P is reset 0 the RAM is addressed by AIPR address counter and the Ai operand is read from the RAM.

SYMBOL / PARAMETER	FUNCTION
CMD.5	
XE2P = input Xi operand from the EEPROM	set and reset by the CPU. When the XE2P bit is set 1 the EEPROM is addressed by XIPE (EADRL2) address pointer and the Xi operand is read from the EEPROM. When XE2P is reset 0 the RAM is addressed by XIPR address counter and the Xi operand is read from the RAM.
CMD.4	
OXOR = output exclusive OR	set and reset by the CPU. When the OXOR bit is set 1 the output data value is: Xi ⊕ calculation result. When the OXOR bit is reset 0 the output data value is: CXOR register content ⊕ calculation result. If the output value must be the real calculation result: OXOR bit is reset 0 and the CXOR register content = 00H.
CMD.3	
SHIFT = control of Ao result shift	set and reset by the CPU. When the SHIFT bit is set 1 the Ao result consists of four registers Ao3, Ao2, Ao1, Ao0, thus the output data is delayed four times which leads to a 32-bit result shift. If the SHIFT bit is reset 0, the output pipeline has only one register Ao3. The result is not shifted.
CMD.2	
LDRAM = Load A3, A2, A1, A0 registers from the RAM	set and reset by the CPU. When the LDRAM bit is set 1 the calculation unit reloads A3, A2, A1, A0 registers from the RAM at the start of the computation. When the LDRAM bit is reset 0, the A3, A2, A1, A0 registers are reloaded from the Ao3, Ao2, Ao1, Ao0 output pipeline registers.
CMD.1	
M32 = 32-bit operands	set and reset by the CPU. When the M32 bit is set 1 the multiplier operands are A[3..0] (32-bits), APR is incremented by four. When the M32 bit is reset 0 the multiplier operands are A[2-0] (28-bits), APR is incremented by three.
CMD.0	
CMRD = command ready	set by the CPU and cleared by the calculation unit (see Fig. 13). To start a calculation, the CMRD bit is set 1 by the CPU. The CMRD bit will be reset 0 by hardware immediately after beginning the calculation. The CMD register is then ready to take the next command word from the CPU. When the CMRD bit is set 1 by the CPU while the calculation unit is still active, the calculation unit does not stop at the end of the current computation. The new calculation starts immediately with the new command parameters. In this case there will be no interrupt request.

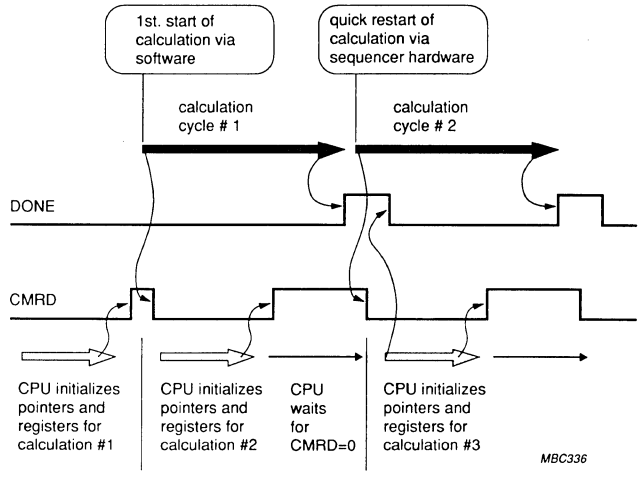


Fig.13 CPU and calculation unit timings.

Table 10 CMDSTAT SFR

SFR Command and Status register (SFR) loaded and read by the CPU →	4	3	2	1	0
	DRA	DRX	RUN	CCY	DONE
	↓	↓	↑	↑	↑
	to the sequencer		from the calculation unit		

Table 11 Description of the CMDSTAT bits

SYMBOL / PARAMETER	FUNCTION
CMDSTAT bits 7, 6, 5, not applicable	
CMDSTAT.4	
DRA = READ disable of the operand Ai	set and reset by the CPU. When the DRA bit is set 1, READ operation of the operand Ai is disabled and Ai is cleared. In this mode the AIPR address counter is not decremented.
CMDSTAT.3	
DRX = READ disable of the operand Xi	set and reset by the CPU. When the DRX bit is set 1, READ operation of the operand Xi is disabled and Xi is cleared. In this mode the XIPR address counter is not decremented.
CMDSTAT.2	
RUN = active calculation unit	set and reset by the calculation unit. While the calculation unit is active, the RUN bit is set 1, otherwise the RUN bit is reset 0 (cleared).

SYMBOL / PARAMETER	FUNCTION
CMDSTAT bits 7, 6, 5, not applicable	
CMDSTAT.1	
CCY = carry	set and reset by the calculation unit. When the contents of Ao3, Ao2, Ao1, Ao0 output pipeline registers are greater than 0 (zero), the CCY bit is set 1, otherwise CCY is reset 0.
CMDSTAT.0	
DONE = end of the computation	set by the calculation unit, reset by the CPU. The DONE bit is set 1 by the calculation unit at the end of computation. When set 1, an interrupt request is generated. The DONE flag has to be reset 0 by the CPU during an interrupt service routine.

Cycle counter CNTCYCL

This is an 8-bit register with associated down counter. It counts the number of bytes of the result which the calculation unit shall carry out. The first byte of the result is always zero.

The CPU has WRITE access to the register and READ access to the down counter. CNTCYCL is cleared to 00H during reset. As long as the calculation unit is in standby, the down counter is continuously loaded with the register contents. After start of calculation, the down counter becomes separated from the register and starts counting. The register can then be reloaded by the CPU for the next calculation cycle. As soon as the down counter reaches the state 00H, the calculation cycle is terminated.

Limit registers

The calculation unit has two 8-bit limit registers (SFRs) WRLIM and RDLIM. WRLIM controls the start of Ao result output to RAM, while RDLIM controls the length of Ai and Xi input operands. RDLIM is split into two 4-bit registers. RDLIM (7-4) carry the READ limit for the Xi operand and RDLIM (3-0) the READ limit for Ai.

The contents of WRLIM, RDLIM (7-4), RDLIM (3-0) are compared to the contents of the Cycle Counter CNTCYCL during calculation. As Xi and Ai limits are both only 4-bits wide, these values are expanded to 8-bits for comparison by adding four leading zeros.

WRITE LIMIT REGISTER WRLIM:

- as long as the CNTCYCL contents are greater than the WRLIM's contents, writing of the Ao output result to RAM is inhibited.

READ LIMIT REGISTER RDLIM:

- READ Xi limit (upper 4-bits): when the CNTCYCL contents has reached the Xi limit, the READ value of the Xi operand is 0.
- READ Ai limit (lower 4-bits): when CNTCYCL contents has reached the Ai limit, the READ value of the Ai operand is 0.

The limit registers can be read and written by the CPU. Because of their pipeline structure, they can be re-loaded whilst the calculation unit is active. Both registers are cleared to 00H during reset.

CXOR register

This is an 8-bit wide SFR that provides one operand for an exclusive-OR operation on the calculation result. It can be read and written by the CPU, the default value after reset is 0.

Calculation unit interrupt

At the end of a calculation cycle, an interrupt request is generated (DONE = 1). The DONE flag has to be reset by software during the interrupt service routine. The calculation unit's operation cannot be interrupted by any other interrupt. Interrupt requests are pending until the end of the calculation cycle. They will be acknowledged during the interrupt service routine.

Parallel operation of CPU and calculation unit

The performance of the calculation unit degrades if pointer and control register initializations are done in between two consecutive calculation cycles. The full calculation speed is reached when these initializations are carried out by the CPU in parallel to a computation

cycle of the calculation unit (see Fig. 13). Thus with parallel initializations, when a current computation cycle is completed, all of the required initializations have already been done to enable the next computation cycle of the calculation unit to start immediately.

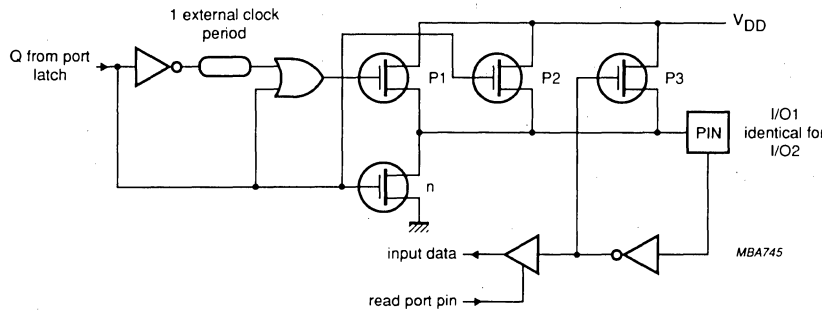
IO register

The 83C852 has 2 I/O lines: I/O1 and I/O2. Line I/O1 is represented by IO1 (bit 0) and line I/O2 is represented by IO2 (bit 1) of the IO register (SFR). IO bits: 7, 6, 5, 4, 3 and 2 are don't care.

Table 12 I/O SFR

–	–	–	–	–	–	IO2	IO1
7	6	5	4	3	2	1	0

Either I/O line can be used independently from the other as an input or as an output. For an I/O line to be used as an input, a set 1 must first be written to it's port-latch. See Fig. 14. The strong output driver FET P1 is turned off after one external clock period. The pin is then pulled HIGH by the weak pull-up FETs P2 and P3. It can be pulled LOW by an external source. After a hardware reset, both port latches contain a set 1 and both lines I/O1 and I/O2 are in Input mode.



for use in ISO standard half duplex serial communication, only I/O1 is needed. Full duplex serial communication may be carried out via both lines I/O1 and I/O2.

Fig.14 Input/Output buffer (I/O1 and I/O2).

Timers

The 83C852 has two 16-bit timer registers Timer 0 and Timer 1. The timer registers are incremented each machine cycle and are thus capable of counting machine cycles. Since a machine cycle consists of 6 external clock periods, the count rate is 1/6 of the clock frequency. Each timer has three operating modes:

- Mode 0 = 13-bit timer
- Mode 1 = 16-bit timer
- Mode 2 = 8-bit timer with auto-reload.

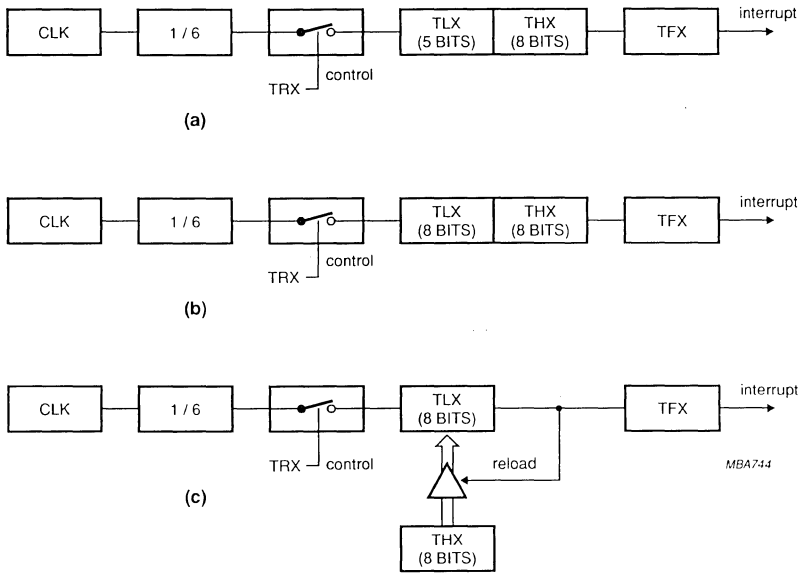


Fig.15 83C852 Timers

Timers 0 and 1 are controlled via the two SFRs: Timer Mode Control (TMOD) and Timer Control/External Interrupt Control (TCON).

Table 13 TMOD SFR

TIMER 1				TIMER 2			
–	–	**M1**	**M0**	–	–	**M1**	**M0**
7	6	5	4	3	2	1	0

Table 14 Description of the TMOD bits

SYMBOL	PARAMETER	FUNCTION
Timer 1		
–	TMOD.7	reserved, don't care
–	TMOD.6	reserved, don't care
M1	TMOD.5	Timer 1 mode select
M0	TMOD.4	Timer 1 mode select
Timer 0		
–	TMOD.3	reserved, don't care
–	TMOD.2	reserved, don't care
M1	TMOD.1	Timer 0 mode select
M0	TMOD.0	Timer 0 mode select
M1 and M0 operating modes		
0	0	8-bit timer "THx" with "TLx" as 5-bit prescaler
0	1	16-bit timer "THx" and "TLx" are cascaded. There is no prescaler
1	0	8-bit auto-reload timer "THx" holds a value which is reloaded into "TLx" each time it overflows.

Table 15 TCON SFR

TIMER CONTROL				EXTERNAL INTERRUPT CONTROL			
TF1	**TR1**	**TF0**	**TR0**	**–**	**IOSW**	**IE0**	**IT0**
7	6	5	4	3	2	1	0

Table 16 Description of the TCON bits

SYMBOL	PARAMETER	FUNCTION
Timer control		
TF1	TCON.7	Timer 1 overflow flag. Set by hardware on Timer 1 overflow. Cleared by hardware when processor vectors to interrupt routine.
TR1	TCON.6	Timer 1 run control bit. Set/cleared by software to turn Timer 1 ON/OFF.
TF0	TCON.5	Timer 0 overflow flag. Set by hardware on Timer 0 overflow. Cleared by hardware when processor vectors to interrupt routine.
TR0	TCON.4	Timer 0 run control bit. Set/cleared by software to turn Timer 0 ON/OFF.
External Interrupt Control		
–	TCON.3	reserved, don't care.
IOSW	TCON.2	switch for external interrupt source: 0 = I/O1 is used as external interrupt source; 1 = I/O2 is used is used as external interrupt source.
IE0	TCON.1	is the external interrupt 0 edge flag. If IT0 is set 1, the IE0 bit is set 1 by hardware when the external interrupt source (either I/O1 pin or I/O2 pin) is detected to have made a 1 to 0 transition. The IE0 bit is cleared by hardware when the processor transfers control to the interrupt service routine.
IT0	TCON.0	determines whether external interrupt is edge-triggered or level-triggered. If IT0 is set 1: external interrupt 0 is edge-triggered. If IT0 is reset 0, external interrupt 0 is triggered by a detected LOW at the external interrupt source.

Interrupt system

The 83C852 has five interrupt sources, each can be programmed to one of two priority interrupt levels, either HIGH or LOW. The five interrupt sources are listed below:

1. I/O: external request from either I/O line
2. Timer 0: overflow from Timer 0
3. Cell: end of calculation
4. Timer 1: overflow from Timer 1
5. EEPROM: completion of EEPROM programming.

Each interrupt source can be individually enabled or disabled, all interrupt sources can also be globally enabled or disabled.

Each interrupt source can be programmed to either a HIGH or a LOW priority interrupt level. A LOW can be interrupted by a HIGH priority interrupt, but not by another LOW priority interrupt. A HIGH priority interrupt cannot be interrupted.

Only one of the I/O lines (either I/O1 or I/O2) can be used as an external interrupt source at a time. This selection is made by IOSW bit from TCON register.

Interrupt vectors

The microcontroller acknowledges a request from an interrupt by a hardware subroutine call. It pushes the contents of the PC (program counter) into the stack, but it does not save the PSW (program status word). PC is reloaded with an address that depends on the source of the interrupt request, as shown below:

Address	Source
0003H	I/O1 or I/O2
000BH	Timer 0 Overflow
0013H	end of calculation
001BH	Timer 1 Overflow
0023H	completion of EEPROM programming.

Interrupt registers IE and IP

INTERRUPT ENABLE REGISTER IE

Each source can be individually enabled or disabled by setting or clearing the corresponding bit inside the SFR Interrupt Enable register IE. All interrupt sources can also be globally enabled or disabled.

Table 17 IE SFR

EA	–	–	EE	ET1	EC	ET0	EX0
7	6	5	4	3	2	1	0

Table 18 Description of the IE bits

SYMBOL	PARAMETER	FUNCTION
EA	IE.7	general enable/disable control 0 = no interrupt is enabled 1 = any individually enabled interrupt will be accepted
–	IE.6	reserved, don't care
–	IE.5	reserved, don't care
EE	IE.4	enable EEPROM interrupt
ET1	IE.3	enable Timer 1 interrupt
EC	IE.2	enable calculation unit interrupt
ET0	IE.1	enable Timer 0 interrupt
EX0	IE.0	enable external 0 interrupt (from I/O)

Table 19 IP SFR

–	–	–	PE	PT1	PCU	PT0	PX0
7	6	5	4	3	2	1	0

Table 20 Description of the IP bits

SYMBOL	PARAMETER	FUNCTION
–	IP.7	reserved, don't care
–	IP.6	reserved, don't care
–	IP.5	reserved, don't care
PE	IP.4	EEPROM interrupt level
PT1	IP.3	Timer 1 interrupt level
PCU	IP.2	calculation unit interrupt level
PT0	IP.1	Timer 0 interrupt level
PX0	IP.0	external 0 interrupt level

INTERRUPT PRIORITY REGISTER IP

The interrupt level is selected within the SFR Interrupt Priority register IP. Setting a bit to '1' selects HIGH priority.

Hardware security

OPERATING MODE

The microcontroller has two operating modes:

- User mode
- Test mode

The test mode is permanently disabled once the test has been performed.

Low frequency sensor

The low frequency detector circuit triggers a reset of the CPU when the clock frequency falls below f_{CLK} minimum (approximately 500 kHz). When the clock frequency rises above f_{CLK} minimum the reset is de-activated.

Power ON/OFF reset

The power ON/OFF reset circuit triggers a reset of the CPU when the power supply falls below V_{DD} minimum (approximately 3.5 V). When the power supply rises above V_{DD} minimum, this reset is de-activated. When the power-down mode is active (PD is set 1) the power ON/OFF reset is de-activated.

Idle mode and power-down mode

IDLE MODE

The 83C852 provides two power saving operational modes, the idle mode and the power-down mode. In the idle mode, the CPU enters a sleep routine whilst some of the on-chip peripherals (timers and interrupt system) remain active. The contents of the RAM and SFRs remain unchanged during the duration of an idle mode. The idle mode can be terminated by an enabled interrupt or by a hardware reset. Besides stopping the CPU, the idle mode terminates EEPROM write operations and stops operation of the calculation unit.

POWER-DOWN MODE

In the power-down mode, all on-chip internal clocks are frozen. The CPU and all on-chip peripherals stop working. The only exit from a power-down mode is by a hardware reset. The on-chip RAM and SFRs retain their values until the power-down mode is terminated. Reset redefines the SFRs, but does not change the RAM contents. The V_{DD} supply can be reduced to 2 V whilst the power-down mode is active. Both modes are activated by software via the SFR Power Control register PCON. PCON is not bit addressable.

Table 21 PCON SFR

–	–	–	–	GF1	GF0	PD	IDL
7	6	5	4	3	2	1	0

Table 22 Description of the PCON bits

SYMBOL	PARAMETER	FUNCTION
–	PCON.7	reserved, don't care
–	PCON.6	reserved, don't care
–	PCON.5	reserved, don't care
–	PCON.4	reserved, don't care
GF1	PCON.3	general purpose flag bit
GF0	PCON.2	general purpose flag bit
PD	PCON.1	enter power-down mode when set; note 1
IDL	PCON.0	enter idle mode when set; note 1

Note

1. If a logic 1 is written to PD and IDL at the same time, PD takes precedence.

SFRs memory mapping

Table 23

The 83C852 has the following 33 Special Function Registers (SFRs) available to the user.

SFRs ADDRESS	SYMBOL	RESET VALUE	FUNCTION
FBH	WRLIM	0000.0000B	WRITE limit register for calculation unit
FAH	RDLIM	0000.0000B	READ limit for calculation unit
F9H	CNTCYCL	0000.0000B	cycle counter for calculation unit
F7H	ECNTRL2	XXXX.0000B	EEPROM control register (test modes)
F6H	ECNTRL1	0000.0000B	EEPROM control register (user modes)
F5H	ETIM	0000.1000B	EEPROM timer register
F4H	EDAT	XXXX.XXXXB	EEPROM data register
F3H	EADRH	1000.0000B	EEPROM address register HIGH
F2H	EADRL1	0000.0000B	EEPROM address register 1 LOW, address pointer AIPE for calculation unit
F1H	EADRL2	0000.0000B	EEPROM address register 2 LOW, address pointer XIPE for calculation unit
F0H	B	0000.0000B	B register
E0H	ACC	0000.0000B	accumulator
D0H	PSW	0000.0000B	program status word
B8H	IP	XXX0.0000B	interrupt priority register
B0H	IO	XXXX.XX11B	I/O register
A8H	IE	0XX0.0000B	interrupt enable register
A7H	AOPR	0000.0000B	AOPR register for calculation unit
A6H	APR	0000.0000B	APR register for calculation unit
A5H	XIPR	0000.0000B	XIPR register for calculation unit
A4H	AIPR	0000.0000B	AIPR register for calculation unit
A3H	CXOR	0000.0000B	CXOR register for calculation unit
99H	CMD	0000.0000B	command register for calculation unit
98H	CMDSTAT	XXX0.0000B	command and status register for calculation unit
8DH	TH1	0000.0000B	Timer 1 HIGH
8CH	TH0	0000.0000B	Timer 0 HIGH
8BH	TL1	0000.0000B	Timer 1 LOW
8AH	TL0	0000.0000B	Timer 0 LOW
89H	TMOD	XX00.XX00B	Timer 0 and 1 mode control
88H	TCON	0000.X000B	Timer 0 and 1 control and external interrupt control
87H	PCON	XXXX.0000B	power control register
83H	DPH	0000.0000B	data pointer HIGH
82H	DPL	0000.0000B	data pointer LOW
81H	SP	0000.0111B	stack pointer

	← Bit addressable			8 Bytes					
F8		CNTCYCL	RDLIM	WRLIM					FF
F0	B	EADRL2	EADRL1	EADRH	EDAT	ETIM	ECNTRL1	ECNTRL2	F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8									CF
C0									C7
B8	IP								BF
B0	IO								B7
A8	IE								AF
A0				CXOR	AIPR	XIPR	APR	AOPR	A7
98	CMDSTAT	CMD							9F
90									97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80		SP	DPL	DPH				PCON	87

MBA746

83C852 specific SFR's

Fig.16 SFRs mapping.

INSTRUCTION SET

The instruction set consists of 49 single-byte, 46 two-byte and 16 three-byte instructions. When using a 6 MHz external clock, 64 instructions execute in 1 cycle (1 μs) and 45 instructions execute in 2 cycles (2 μs). Multiply and divide instructions execute in 4 cycles (4 μs).

INSTRUCTION SET DESCRIPTION

For data addressing modes, Hexadecimal opcode cross-reference and invalid instructions, see Table notes.

MNEMONIC		DESCRIPTION	BYTES	CYCLES	OPCODE (HEX)
Arithmetic operation					
ADD	A,Rr	Add register to A	1	1	2*
ADD	A,direct	Add direct byte to A	2	1	25
ADD	A,@Ri	Add indirect RAM to A	1	1	26, 27
ADD	A,#data	Add immediate data to A	2	1	24
ADDC	A,Rr	Add register to A with carry flag	1	1	3*
ADDC	A,direct	Add direct byte to A with carry flag	2	1	35
ADDC	A,@Ri	Add indirect RAM to A with carry flag	1	1	36, 37
ADDC	A,#data	Add immediate data to A with carry flag	2	1	34
SUBB	A,Rr	Subtract register from A with borrow	1	1	9*
SUBB	A,direct	Subtract direct byte from A with borrow	2	1	95
SUBB	A,@Ri	Subtract indirect RAM from A with borrow	1	1	96, 97
SUBB	A,#data	Subtract immediate data from A with borrow	2	1	94
INC	A	Increment A	1	1	04
INC	Rr	Increment register	1	1	0*
INC	direct	Increment direct byte	2	1	05
INC	@Ri	Increment indirect RAM	1	1	06, 07
DEC	A	Decrement A	1	1	14
DEC	Rr	Decrement register	1	1	1*
DEC	direct	Decrement direct byte	2	1	15
DEC	@Ri	Decrement indirect RAM	1	1	16, 17
INC	DPTR	Increment data pointer	1	2	A3
MUL	AB	Multiply A & B	1	4	A4
DIV	AB	Divide A by B	1	4	84
DA	A	Decimal adjust A	1	1	D4

Instruction set description (continued)

MNEMONIC		DESCRIPTION	BYTES	CYCLES	OPCODE (HEX)
Logic operations					
ANL	A,Rr	AND register to A	1	1	5*
ANL	A,direct	AND direct byte to A	2	1	55
ANL	A,@Ri	AND indirect RAM to A	1	1	56, 57
ANL	A,#data	AND immediate data to A	2	1	54
ANL	direct,A	AND A to direct byte	2	1	52
ANL	direct,#data	AND immediate data to direct byte	3	2	53
ORL	A,Rr	OR register to A	1	1	4*
ORL	A,direct	OR direct byte to A	2	1	45
ORL	A,@Ri	OR indirect RAM to A	1	1	46, 47
ORL	A,#data	OR immediate data to A	2	1	44
ORL	direct,A	OR A to direct byte	2	1	42
ORL	direct,#data	OR immediate data to direct byte	3	2	43
XRL	A,Rr	Exclusive-OR register to A	1	1	6*
XRL	A,direct	Exclusive-OR direct byte to A	2	1	65
XRL	A,@Ri	Exclusive-OR indirect RAM to A	1	1	66, 67
XRL	A,#data	Exclusive-OR immediate data to A	2	1	64
XRL	direct,A	Exclusive-OR A to direct byte	2	1	62
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	2	63
CLR	A	Clear A	1	1	E4
CPL	A	Complement A	1	1	F4
RL	A	Rotate A left	1	1	23
RLC	A	Rotate A left through the carry flag	1	1	33
RR	A	Rotate A right	1	1	03
RRC	A	Rotate A right through the carry flag	1	1	13
SWAP	A	Swap nibbles within A	1	1	C4

Instruction set description (continued)

MNEMONIC		DESCRIPTION	BYTES	CYCLES	OPCODE (HEX)
Data transfer					
MOV	A,Rr	Move register to A	1	1	E*
MOV	A,direct**	Move direct byte to A	2	1	E5
MOV	A,@Ri	Move indirect RAM to A	1	1	E6, E7
MOV	A,#data	Move immediate data to A	2	1	74
MOV	Rr,A	Move A to register	1	1	F*
MOV	Rr,direct	Move direct byte to register	2	2	A*
MOV	Rr,#data	Move immediate data to register	2	1	7*
MOV	direct,A	Move A to direct byte	2	1	F5
MOV	direct,Rr	Move register to direct byte	2	2	8*
MOV	direct,direct	Move direct byte to direct	3	2	85
MOV	direct,@Ri	Move indirect RAM to direct byte	2	2	86, 87
MOV	direct,#data	Move immediate data to direct byte	3	2	75
MOV	@RI,A	Move A to indirect RAM	1	1	F6, F7
MOV	@Ri,direct	Move direct byte to indirect RAM	2	2	A6, A7
MOV	@Ri,#data	Move immediate data to indirect RAM	2	1	76, 77
MOV	DPTR,#data 16	Load data pointer with a 16-bit constant	3	2	90
MOVC	A,@A+DPTR	Move code byte relative to DPTR to A	1	2	93
MOVC	A,@A+PC	Move code byte relative to PC to A	1	2	83
PUSH	direct	Push direct byte onto stack	2	2	C0
POP	direct	Pop direct byte from stack	2	2	D0
XCH	A,Rr	Exchange register with A	1	1	C*
XCH	A,direct	Exchange direct byte with A	2	1	C5
XCH	A,@Ri	Exchange indirect RAM with A	1	1	C6, C7
XCHD	A,@Ri	Exchange LOW-order digit indirect RAM with A	1	1	D6, D7
Note: the following MOVX instructions of 80C51 set are not applicable to 83C852					
MOVX	A,@Ri	Move external RAM (8-bit address) to A	1	2	E2, E3
MOVX	A,@DPTR	Move external RAM (16-bit address) to A	1	2	E0
MOVX	@Ri,A	Move A to external RAM (8-bit address)	1	2	F2, F3
MOVX	@DPTR,A	Move A to external RAM (16-bit address)	1	2	F0

Instruction set description (continued)

MNEMONIC		DESCRIPTION	BYTES	CYCLES	OPCODE (HEX)
Boolean variable manipulation					
CLR	C	Clear carry flag	1	1	C3
CLR	bit	Clear direct bit	2	1	C2
SETB	C	Set carry flag	1	1	D3
SETB	bit	Set direct bit	2	1	D2
CPL	C	Complement carry flag	1	1	B3
CPL	bit	Complement direct bit	2	1	B2
ANL	C,bit	AND direct bit to carry flag	2	2	82
ANL	C,/bit	AND complement of direct bit to carry flag	2	2	B0
ORL	C,bit	OR direct bit to carry flag	2	2	72
ORL	C,/bit	OR complement of direct bit to carry flag	2	2	A0
MOV	C,bit	Move direct bit to carry flag	2	1	A2
MOV	bit,C	Move carry flag to direct bit	2	2	92
Program and machine control					
ACALL	addr11	Absolute subroutine call	2	2	•1addr
LCALL	addr16	Long subroutine call	3	2	12
RET		Return from subroutine	1	2	22
RETI		Return from interrupt	1	2	32
AJMP	addr11	Absolute jump	2	2	♦1addr
LJMP	addr16	Long jump	3	2	02
SJMP	rel	Short jump (relative address)	2	2	80
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	2	73
JZ	rel	Jump if A is zero	2	2	60
JNZ	rel	Jump if A is not zero	2	2	70
JC	rel	Jump if carry flag is set	2	2	40
JNC	rel	Jump if carry flag is not set	2	2	50
JB	bit,rel	Jump if direct bit is set	3	2	20
JNB	bit,rel	Jump if direct bit is not set	3	2	30
JBC	bit,rel	Jump if direct bit is set and clear bit	3	2	10
CJNE	A,direct,rel	Compare direct to A and jump if not equal	3	2	B5
CJNE	A,#data,rel	Compare immediate to A and jump if not equal	3	2	B4
CJNE	Rr,#data,rel	Compare immed. to reg. and jump if not equal	3	2	B*
CJNE	@Ri,#data,rel	Compare immed. to ind. and jump if not equal	3	2	B6, B7
DJNZ	Rr,rel	Decrement register and jump if not zero	2	2	D*
DJNZ	direct,rel	Decrement direct and jump if not zero	3	2	D5
NOP		No operation	1	1	00

NOTES TO INSTRUCTION SET TABLE

MNEMONIC	DESCRIPTION
Data addressing modes	
Rr	working register R0-R7.
direct	128 internal RAM locations and any special function register (SFR).
@Ri	indirect internal RAM location addressed by register R0 or R1 of the actual register bank.
#data	8-bit constant included in instruction.
#data 16	16-bit constant included as bytes 2 and 3 of instruction.
bit	direct addressed bit in internal RAM or SFR.
addr16	16-bit destination address. Used by LCALL and LJMP. The branch will be anywhere within the 64K byte program memory address space.
addr11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2K byte page of program memory as the first byte of the following instruction.
rel	Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is −128 to +127 bytes relative to first byte of the following instruction.
Hexadecimal opcode cross-reference	
*	8, 9, A, B, C, D, E, F.
•	11, 31, 51, 71, 91, B1, D1, F1.
♦	01, 21, 41, 61, 81, A1, C1, E1.
Invalid instructions:	
note **	MOV A, ACC is not a valid instruction.
MOVX	MOVX instructions of 80C51 set are not applicable to 83C852.

INSTRUCTION MAP

MOVX instructions not applicable

first hexadecimal character of opcode ↓ / second hexadecimal character of opcode →

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	AJMP addr11	LJMP addr16	RR A	INC A	INC dir	INC @Ri 0	1	INC Rr 0	1	2	3	4	5	6	7
1	JBC bit,rel	ACALL addr11	LCALL addr16	RRC A	DEC A	DEC dir	DEC @Ri 0	1	DEC Rr 0	1	2	3	4	5	6	7
2	JB bit,rel	AJMP addr11	RET	RL A	ADD A,#data	ADD A,dir	ADD A,@Ri 0	1	ADD A,Rr 0	1	2	3	4	5	6	7
3	JNB bit,rel	ACALL addr11	RETI	RLC A	ADDC A,#data	ADDC A,dir	ADDC A,@Ri 0	1	ADDC A,Rr 0	1	2	3	4	5	6	7
4	JC rel	AJMP addr11	ORL dir,A	ORL dir,#data	ORL A,#data	ORL A,dir	ORL A,@Ri 0	1	ORL A,Rr 0	1	2	3	4	5	6	7
5	JNC rel	ACALL addr11	ANL dir,A	ANL dir,#data	ANL A,#data	ANL A,dir	ANL A,@Ri 0	1	ANL A,Rr 0	1	2	3	4	5	6	7
6	JZ rel	AJMP addr11	XRL dir,A	XRL dir,#data	XRL A,#data	XRL A,dir	XRL A,@Ri 0	1	XRL A,Rr 0	1	2	3	4	5	6	7
7	JNZ rel	ACALL addr11	ORL C,bit	JMP @A+DPTR	MOV A,#data	MOV dir,#data	MOV @Ri,#data 0	1	MOV Rr,#data 0	1	2	3	4	5	6	7
8	SJMP rel	AJMP addr11	ANL C,bit	MOVC A,@A+PC	DIV AB	MOV dir,dir	MOV dir,@Ri 0	1	MOV dir,Rr 0	1	2	3	4	5	6	7
9	MOV DPTR, #data	ACALL addr11	MOV bit,C	MOVC A,@A+DPTR	SUBB A,#data	SUBB A,dir	SUBB A,@Ri 0	1	SUBB A,Rr 0	1	2	3	4	5	6	7
A	ORL C,/bit	AJMP addr11	MOV C,bit	INC DPTR	MUL AB		MOV @Ri,dir 0	1	MOV Rr,dir 0	1	2	3	4	5	6	7
B	ANL C,/bit	ACALL addr11	CPL bit	CPL C	CJNE A, #data,rel	CJNE A,dir,rel	CJNE @Ri,#data,rel 0	1	CJNE Rr,#data,rel 0	1	2	3	4	5	6	7
C	PUSH dir	AJMP addr11	CLR bit	CLR C	SWAP A	XCH A,dir	XCH A,@Ri 0	1	XCH A,Rr 0	1	2	3	4	5	6	7
D	POP dir	ACALL addr11	SETB bit	SETB C	DA A	DJNZ dir,rel	XCHD A,@Ri 0	1	DJNZ Rr,rel 0	1	2	3	4	5	6	7
E	MOVX A,@DPTR	AJMP addr11	MOVX A,@Ri 0	1	CLR A	MOV * A,dir	MOV A,@Ri 0	1	MOV A,Rr 0	1	2	3	4	5	6	7
F	MOVX @DPTR,A	ACALL addr11	MOVX @Ri,A 0	1	CPL A	MOV dir,A	MOV @Ri,A 0	1	MOV Rr,A 0	1	2	3	4	5	6	7

* MOV A,ACC is not a valid instruction.

ISO INFORMATION

The following ISO characteristics information as applicable to this data sheet may be superseded. Please ensure that the latest version of ISO information is studied for relevant features.

ISO ELECTRICAL CHARACTERISTICS

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
I/O						
V_{IH}	input voltage HIGH	$I_{IH\ (max.)}$ = ±20 µA	0.7 x V_{DD}	–	V_{DD} +0.3	V
V_{IL}	input voltage LOW	$I_{IL\ (max.)}$ = −1 mA	−0.3	–	0.8	V
V_{OH}	output voltage HIGH	$I_{OH\ (max.)}$ = −20 µA; note 1	3.8	–	V_{DD}	V
V_{OL}	output voltage LOW	$I_{OL\ (max.)}$ = +1 mA	0	–	0.4	V
C_{VO}	input/output pin capacitance		–	–	30	pF
$t_{r/f}$	I/O rise/fall times	C_{IN} = 30 pF; C_{OUT} = 30 pF	–	–	1	µs
CLK						
V_{IH}	input voltage HIGH	$I_{IH\ (max.)}$ = ±20 µA	0.7 x V_{DD}	–	V_{DD} +0.3	V
V_{IL}	input voltage HIGH	$I_{IL\ (max.)}$ = ±200 µA	−0.3	–	0.5	V
C_I	input pin capacitance		–	–	30	pF
$t_{r/f}$	CLK rise/fall times	C_{IN} = 30 pF	–	–	9% of period with a max. of 0.5 µs	µs
$\overline{\text{RESET}}$						
V_{IH}	input voltage HIGH	$I_{IH\ (max.)}$ = ±20 µA	0.7 x V_{DD}	–	V_{DD} +0.3	V
V_{IL}	input voltage LOW	$I_{IL\ (max.)}$ = ±200 µA	−0.3	–	0.5	V

Note

1. It is assumed that a pull-up resistor is used in the interface device (recommend value = 20 kΩ).

LIMITING VALUES

In accordance with the Absolute Maximum System (IEC 134)

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_I	input voltage on any pin with respect to ground (V_{SS})	−0.5	±6.5	V
I_I; I_O	input/output current on I/O1 or I/O2 pin	–	±5	mA
P_{tot}	total power dissipation per package	–	1	W
T_{stg}	storage temperature range	−65	150	°C
T_{amb}	operating ambient temperature range	0	70	°C

HANDLING

Inputs and outputs are protected against electrostatic discharge in normal handling. However, to be totally safe, it is desirable to take precautions appropriate to handling MOS devices (see 'Handling MOS Devices').

CHARACTERISTICS

V_{DD} = 5 V (±10 %); V_{SS} = 0 V; T_{amb} = 0 to 70 °C; all voltages with respect to V_{SS} unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
DC						
V_{DD}	supply voltage range		4.5	–	5.5	V
I_{DD}	supply current	f_{CLK} = 3.57 MHz	–	–	10	mA
I_{DD}	supply current operating mode	f_{CLK} = 6.0 MHz	–	–	15	mA
I_{ID}	supply current idle mode	f_{CLK} = 6.0 MHz	–	–	3	mA
I_{PD}	power-down current	2 V ≤ V_{PD} ≤ V_{DD} max.	–	–	100	µA
I/O1; I/O2						
V_{IL}	input voltage LOW		–0.5		0.2 V_{DD}	V
V_{IH}	input voltage HIGH		0.2 V_{DD} +0.9		V_{DD} +0.5	V
I_{IL}	input current LOW	V_I = +0.45 V	–	–	–50	µA
I_{IL}	input current HIGH-to-LOW		–	–	650	µA
V_{OH}	output voltage HIGH	I_{OH} = –20 µA	3.8	–	–	V
V_{OL}	output voltage LOW	I_{OL}= 1.0 mA	–	–	0.4	V
V_{IL}	$\overline{RESET}$; CLK input voltage LOW		–0.5	–	0.2 V_{DD} –0.1	V
V_{IH}	$\overline{RESET}$; CLK input voltage HIGH		0.7 V_{DD}	–	V_{DD} +0.5	V
I_{LI}	input leakage current ($\overline{RESET}$; CLK)	0.45 < V_I < V_{DD}	–	–	±10	µA
ESD	ESD protection	C = 100 pF; R = 1.5 k	–	–	2.0	kV
AC						
f_{CLK}	external clock frequency	internal operating frequency = f_{CLK}	1	–	6	MHz
t_{CYC}	cycle time		1	–	–	µs
t_{CLK}	clock pulse width		45	–	55	%
t_r	clock rise time		–	–	tbf	ns
t_f	clock fall time		–	–	tbf	ns
t_{POR}	power-on reset delay		tbf	–	tbf	tbf
t_{rw}	reset pulse width		12/ f_{CLK}	–	–	s
t_e	EEPROM ERASE time		–	5.0	–	ms
t_w	EEPROM WRITE time		–	5.0	–	ms
t_s	EEPROM data retention time	T_{amb} = 55° C	10.0	–	–	yrs
$N_{e/w}$	EEPROM endurance (number of erase/write cycles	t_e = 5 ms; t_w = 5 ms	10 000	–	–	cycles
C	I/O1; I/O2; $\overline{RESET}$; f_{CLK} pin capacitance	f_{CLK} = 1 MHz; T_{amb} = 25 °C	–	–	10	pF

Section 5
Application Notes

80C51-Based
8-Bit Microcontrollers

CONTENTS

AN408	80C451 Operation of Port 6	665
AN417	256k Centronics Printer Buffer Using the 87C451 Microcontroller	676
AN418	Counter/Timer 2 of the 83C552 Microcontroller	689
AN420	Using up to 5 External Interrupts on 80C51 Family Microcontrollers	696
AN422	Using the 8XC751 Microcontroller as an I^2C Bus Master	698
AN423	Software Driven Serial Communication Routines for the 83C751 and 83C752 Microcontrollers	716
AN424	8051 Family Warm Boot Determinations	722
AN425	Interfacing the PCD8584 I^2C-bus Controller to 80C51 Family Microcontrollers	724
AN426	Controlling Aire Core Meters with the 87C751 and SA 5775	744
AN427	Timer I in Non-I^2C Applications of the 83/87C751/752 Microcontrollers	759
AN428	Using the ADC and PWM of the 83C752/87C752	765
AN429	Airflow Measurement Using the 83/87C752 and "C"	772
AN431	9XC1XX Philips 16/32-Bit Microcontroller Series	792
AN432	Parallel Port on the SBE 68070	796
AN436	"Opti-Mizer" Power Management for Notebook Computers Using the 8XC752 Microcontroller	802
AN438	I^2C Routines for 8XC528	812
AN442	(BCM) 87C751 Specification for a Bus-Controlled Monitor	850

80C451 operation of port 6

AN408

INTRODUCTION

The features of the 80C451 are shared with the 80C51 or are conventional except for the operation of port 6. The flexibility of this port facilitates high–speed parallel data communications. This application note discusses the use of port 6 and is divided into the following sections:

1. Port 6 as a processor bus interface.
2. Using port 6 as a standard pseudo bidirectional I/O port.
3. Implementation of parallel printer ports.

This information applies to all versions of the part: 80C451, 83C451, and the 87C451.

PORT 6 AS A PROCESSOR BUS INTERFACE

Port 6 allows use of the 80C451 as an element on a microprocessor type bus. The host processor could be a general purpose MPU or the data bus of a microcontroller like the 80C451 itself. This feature allows single or multiple 80C451 controllers to be used on a bus as flexible peripheral processing elements. Applications could include keyboard scanners, serial I/O controllers, servo controllers, etc.

OPERATION

On reset, port 6 is programmed correctly for use as a bus interface (see Figure 2). This prevents the interface from disrupting data on the bus of the host processor during power–up. Software initialization of the CSR (Control Status Register) is not required. A dummy read of port 6 may be required to clear the IBF (Input Buffer Full) flag since it could be set by turn on transients on the bus of the host processor. On reset, the CSR of the 83C451 is programmed to allow the following:

1. AFLAG is an input controlling the port select function. If AFLAG is high, the contents of the CSR is output on port 6 when the port is read by the host. If AFLAG is low, then the contents of the output latch is output when port 6 is read by the host.
2. BFLAG is an input controlling the port enable function. In this mode when BFLAG is high, the input latch and the output drivers are disabled and the flags are not affected by the IDS (Input Data Strobe) or ODS (Output Data Strobe) signals. When BFLAG is low, the port is enabled for reading and writing under the control of IDS and ODS pins.

Figure 1 shows one possible example of an 80C451 on a memory bus. This arrangement allows the main processor to query port 6 for flag status without interrupting the 80C451. If the address decoder, shown in Figure 1, enables port 6 on the 80C451 when the address is 8000H or 8001H, and the address line A0 controls the port select feature, then the host processor can read and write to port 6 using address 8000H. Since the port select function is being controlled by the address line A0, the CSR contents can be read by the host processor at address 8001H.

By testing the CSR contents in this way, the host processor can tell if new data has been written to the port 6 output latch since it last read the port or if the 80C451 has read the last byte that the host wrote to the port. Conversely, the 80C451 can poll the flags in its CSR to see if the host processor has written to or read from port 6 since the last time it serviced the port.

If desired, an interrupt source for the 80C451 can be derived easily from the port enable source as shown by the dashed line in Figure 1.

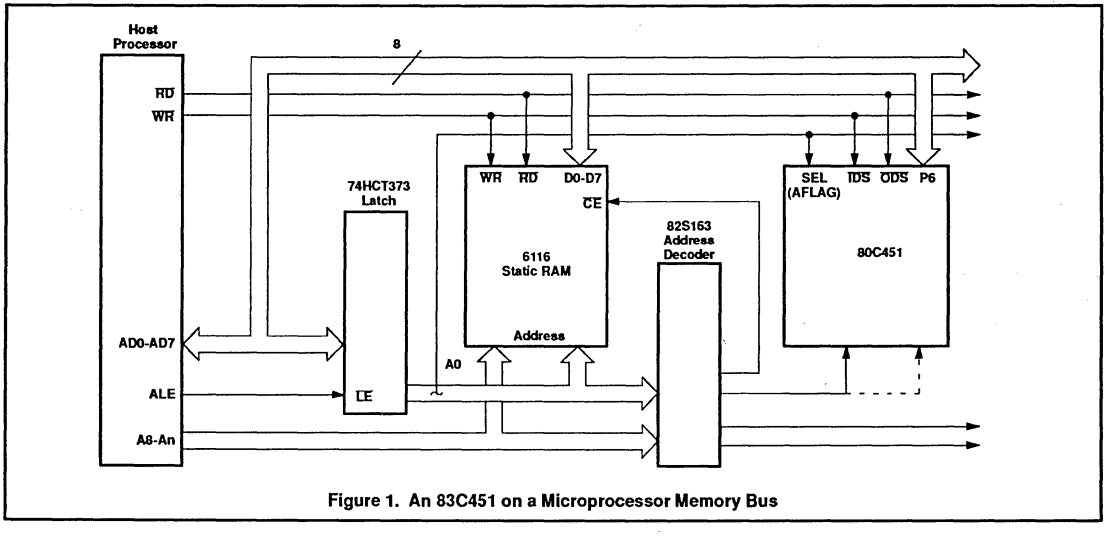


Figure 1. An 83C451 on a Microprocessor Memory Bus

80C451 operation of port 6 AN408

SOFTWARE EXAMPLES

To write to port 6 on the bus shown in Figure 1, the host processor first reads the CSR contents at address 8001H, and tests the input buffer full flag (CSR bit 0). If the flag is clear, the host writes a byte to address 8000H. This loads the input buffer latch of port 6 and sets the input buffer full flag. Conversely, the 80C451 polls the IBF flag and reads a byte from port 6 when it finds the flag set. The flag is automatically reset when this internal read occurs.

80C451 ROUTINE TO READ ONE BYTE FROM HOST VIA PORT 6

```
RCVR:   JNB CSR.0,RCVR   ;TEST IBF FLAG
        MOV A,P6         ;WHEN FLAG IS SET READ BYTE
        RET
```

80C451 ROUTINE TO WRITE ONE BYTE TO THE 83C451 PORT 6

If the host processor is an 80C51, the following routine will write a byte of data to the 80C451. The data involved is passed to the routine through register 1.

```
XMIT:   MOV DPTR,8001H
TEST:   MOVX A,@DPTR     ;READ THE CSR
        JB ACC.0,TEST    ;TEST IBF FLAG
        MOV DPTR,8000H
        MOV A,R1
        MOVX @DPTR,A     ;WRITE DATA TO THE 451
        RET
```

80C451 ROUTINE TO WRITE ONE BYTE TO HOST VIA PORT 6

Routines for data transfer in the opposite direction are similar to the above two. The 80C451 version is given below.

```
XMIT:   JB CSR.1,XMIT    ;TEST OBF FLAG
        MOV P6,A         ;WRITE DATA
        RET
```

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
1	1	1	1	1	1		

Figure 2. CSR Programmed to Allow Port 6 as a Bus Interface

USING PORT 6 AS A STANDARD QUASI-BIDIRECTIONAL I/O PORT

To use port 6 as a common I/O port, all of the control pins are tied to ground (see Figure 3). On hardware reset, bits 2 - 7 in the CSR are set to one. Port operation and electrical characteristics become identical to port 1 on the 80C51 and the 80C451 ports 1, 4, and 5. No software initialization is required.

If desired, AFLAG and BFLAG can be used as outputs while port 6 is operating as a standard quasi-bidirectional I/O port (see Figure 4). In this case, only $\overline{IDS}$ and $\overline{ODS}$ are tied to ground and the CSR is initialized to allow operation of AFLAG and BFLAG as simple outputs (see Figure 5).

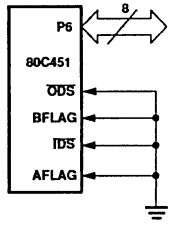


Figure 3. Standard I/O Port on Reset

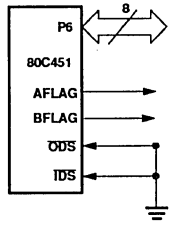


Figure 4. Standard I/O Port on Reset with AFLAG and BFLAG as Outputs

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
1	X	0	X	X	1		

Figure 5. CSR Programmed to Allow AFLAG and BFLAG to Operate as Outputs and Port 6 as a Standard I/O Port

DATA TRANSFER SIGNAL PINS		
Pin No.	Groung Return Pin No.	Signal
1	19	STROBE
2	20	DATA 1
3	21	DATA 2
4	22	DATA 3
5	23	DATA 4
6	24	DATA 5
7	25	DATA 6
8	26	DATA 7
9	27	DATA 8
10	28	ACKNLG
11	29	BUSY

TYPICAL AUXILIARY PIN FUNCTIONS	
Pin No.	Signal
12	PAPER OUT
14	AUTO LINE FEED
16	LOGIC GROUND
17	CHASSIS GND
30	GROUND RETURN
31	RESET PRINTER
32	ERROR
33	GROUND RETURN
36	SLCTIN

Figure 6. Parallel Printer Interface Pin Functions

IMPLEMENTATION OF PARALLEL PRINTER PORTS USING PORT 6

The 80C451 is an excellent choice for a printer controller. The 80C451 has the facilities to permit all of the intelligent features of a common printer to be handled by a single chip:

1. The features of port 6 allow a parallel printer port to be designed with only line driving and receiving chips required as additional hardware.
2. The onboard UART allows RS232 interfacing with only level shifting chips added.
3. The 8-bit parallel ports 0 to 6 are ample to drive onboard control functions, even when ports are used for external memory access, interrupts, and other functions.
4. The RAM addressing ability of ports 0 and 2 can be used to address up to 64k bytes of a hardware buffer/spooler. AFLAG and BFLAG as simple outputs (see Figure 5).
5. The 64k byte ROM addressing capability allows space for the most sophisticated software.

In addition, either end of a parallel interface can be implemented using port 6, and the interfaces can be interrupt driven or polled in either case.

80C451 operation of port 6 AN408

THE INTERFACE

Data transfer on a parallel printer interface occurs across eleven signal lines. The other conductors on the standard plug are used as ground returns or for auxiliary functions (see Figure 6). Only the data transfer signals will be considered.

The Data Transfer Format

The parallel printer interfaces are far more standardized in features than their serial counterpart. However, at least three significant variations exist in handshake style in printers using generic parallel interfaces. This fact influences the design of both port hardware and software. A good transmitter should be able to drive devices with all three styles of handshakes, and a good receiver should generate the handshake most likely compatible with any transmitter.

The Variations

Type 1—Figure 7 shows a common style of handshake and is the style that will be implemented in the receiver examples. A busy signal and an acknowledge strobe pulse are generated for every byte received.

Type 2—Another style of handshake generates a busy signal only when the printer will not be able to accept more data for a relatively long time. Acknowledge pulses are created after every byte received. When the busy signal is generated after a byte is received, the associated acknowledge pulse does not occur until *after* the busy signal returns to logic zero (see Figure 7).

Type 3—A third handshake style does not generate acknowledge pulses, but a busy signal is produced after every byte is received.

PARALLEL PRINTER INTERFACES USING POLLING

Transmitter Operation

This application illustrates the flexibility of the port 6 logic in solving an applications problem. We need to be able to handle all types of acknowledge signals that might be received by the transmitter. We will use the $\overline{ODS}$ pin and output buffer full flag logic to record the receipt of the acknowledge pulse (see Figure 8), but not all parallel receivers generate acknowledge pulses. We could poll the busy signal line, but not all receivers generate busy signals for each byte received; so lack of a busy signal does not imply that we can send another byte. We can, however, expect an acknowledge pulse very shortly after the end of a busy signal if one is going to arrive at all. So we can send a new data byte after having received either a positive transition on the acknowledge line, or shortly after receiving a negative edge on the busy line.

The CSR is programmed to the output only mode. In this mode, the $\overline{ODS}$ pin does not control the output drivers but only the output buffer full flag. The flag serves to record the positive transition of the acknowledge signal. The input latch is not used, but the $\overline{IDS}$ pin is used to set the input buffer full flag. This is used to record the negative transition at the end of the busy signal. Dummy reads by the 80C451 of port 6 will be used to clear the flag. In this example, the AFLAG mode is set only to place the port in the output only mode. The AFLAG pin is not actually used (see Figure 10).

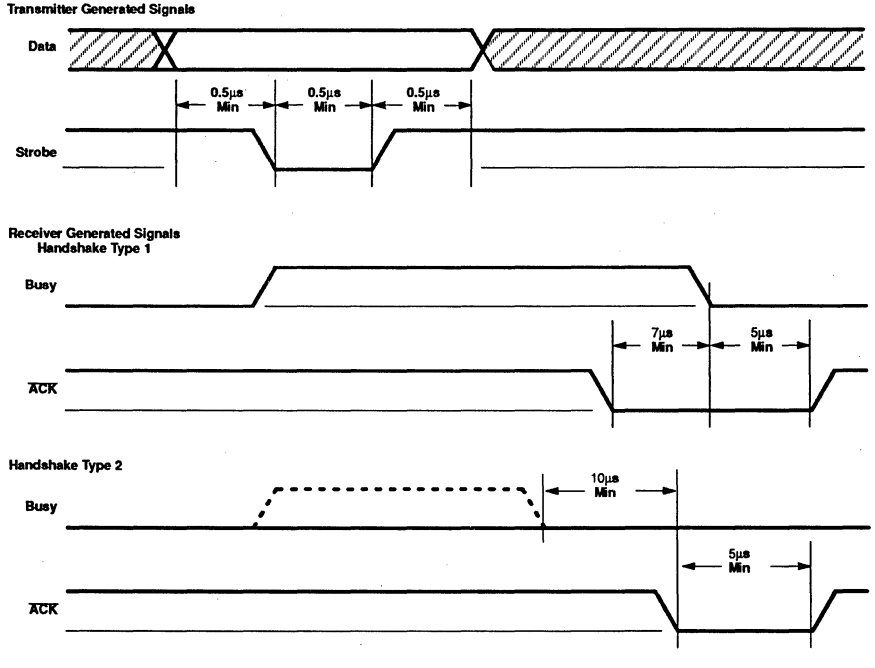


Figure 7. Parallel Printer Interface Signals

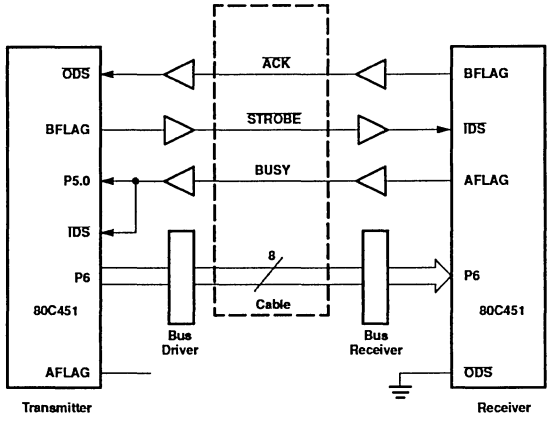


Figure 8. Interconnection for a Parallel Interface Using Polling

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
0	1	1	0	0	1		

Figure 9. CSR Programmed for Polled Transmitter Operation

The transmitter's CSR (control status register) is programmed to the following mode (see Figure 9):

1. CSR bit 6 controls the BFLAG output and therefore the strobe line.
2. The OBF (output buffer full) flag controls the AFLAG output.
3. The OBF is cleared on the positive edge of the ODS input.
4. The IBF flag is cleared on the negative edge of the IDS strobe.

NOTE:
With this combination of modes set, port 6 is in the output only mode.

Receiver Operation

In receiver operation, the IDS input is used to latch in the data transmitted on receipt of the strobe pulse. The receiver's CSR is programmed to allow the following (see Figure 11):

1. The input buffer full flag is output through the BFLAG pin and is used as the busy signal to the transmitter.
2. The IBF flag is set and data is latched on the positive edge of IDS.
3. Writing to the CSR bit 4 controls the AFLAG output and therefore the acknowledge line.

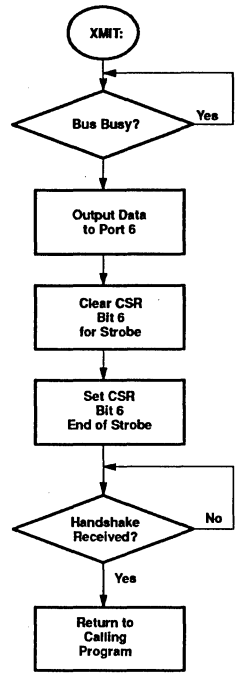


Figure 10. Flow Chart of Polled Parallel Transmitter Operation

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
1	0	0	1	1	0		

Figure 11. CSR Programmed for Polled Parallel Receiver Operation

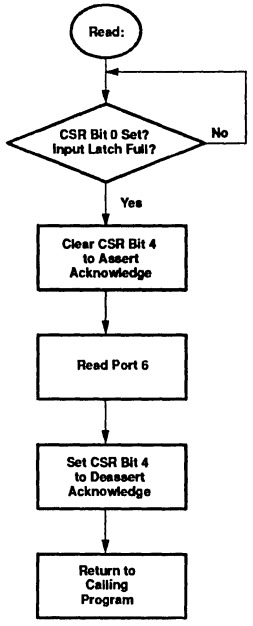


Figure 12. Flow Chart of Polled Parallel Receiver Operation

SOFTWARE EXAMPLES

This polled parallel transmit routine outputs one byte passed to it in the accumulator.

```
P_INIT:   MOV CSR,#064H     ;INITIALIZE PORT 6 OPERATING MODE
P_OUT:    JB P5.0           ;WAIT IF BUSY SIGNAL IS HIGH
          MOV P6,ACC        ;OUTPUT DATA
          MOV R1,P6         ;DUMMY READ TO CLEAR IBF FLAG
          MOV R1,#02H       ;INITIALIZE DELAY COUNTER
          CLEAR CSR.6       ;START STROBE PULSE
          DJNZ R1,$         ;TIME 6 MICROSECOND STROBE PULSE
          SETB CSR.6        ;END STROBE PULSE
WAIT:     JNB CSR.1,OUT     ;EXIT IF ACKNOWLEDGE RCV'D
          JNB CSR.0,WAIT    ;EXIT IF NEGATIVE BUSY EDGE RCV'D
```

This polled parallel receive routine places one byte in the accumulator each time it is called.

```
P_INIT:   MOV CSR,#09CH     ;INITIALIZE PORT 6 OPERATING MODE
          MOV R7,P6         ;DUMMY READ TO CLEAR IBF FLAG
P_IN      JNB CSR.0         ;INPUT BUFFER LATCH FULL?
          CLR CSR.4         ;BEGIN ACKNOWLEDGE PULSE
          MOV R7,#02H       ;INITIALIZE DELAY COUNTER
          DJNZ R7,$         ;TIME ACKNOWLEDGE PULSE
          MOV A,P6          ;READ BYTE – CLEAR BUSY SIGNAL
          MOV R7,#02H       ;INITIALIZE DELAY COUNTER
          DJNZ R7,$         ;TIME ACKNOWLEDGE PULSE
          SETB CSR.4        ;END ACKNOWLEDGE PULSE
          RET
```

INTERRUPT DRIVEN PARALLEL PRINTER INTERFACE (SEE FIGURE 13)

Transmitter Operation

The transmitter's CSR (control status register) is programmed to the following mode (see Figure 14):

1. CSR bit 6 controls the BFLAG output and therefore the strobe line.
2. The OBF (output buffer full) flag controls the AFLAG output.
3. The OBF is cleared on the positive edge of the $\overline{ODS}$ (output data strobe) input.
4. The IBF flag is set on the negative edge of the $\overline{IDS}$ (input data strobe) pin.

NOTE:
With this combination of AFLAG and BFLAG modes set, port 6 is in the output only mode. The output drivers are always enabled and the $\overline{ODS}$ input is only used to clear the OBF flag.

INTO is programmed to be negative edge sensitive and is connected to the OBF flag through the AFLAG pin. The OBF is cleared on the positive edge of $\overline{ODS}$. The net result is that INTO is triggered on the end of the ACK pulse (a positive edge). This signals the transmitter that another byte may be transmitted. The transmitting 83C451 is free to do other tasks prior to this interrupt.

In this routine, Figure 15, the main program establishes a buffer in data memory ended by an ASCII end of text character. To begin outputting the buffer, the routine PSEND is called. The rest of the buffer is emptied by the interrupt vectors to PSEND1.

For printers which generate acknowledge pulses, output rates of 25k transfers per second are achieved. Timer generated interrupts are used to periodically return program execution to the routine to service non-acknowledging printers and to provide a timeout feature. Non-acknowledging printers are serviced at a rate of about 2.5k transfers per second. This maximum rate may be varied by adjusting the timer reload value. As written, the time out procedure attempts to retransmit a byte when the printer has not acknowledged for an excessively long time.

Receiver Operation

In receiver operation, the IDS input is used to latch in the data transmitted on receipt of the strobe pulse. The receiver's CSR is programmed to allow the following (see Figure 16):

1. The input buffer full flag is output through the BFLAG pin and is used as the busy signal to the transmitter. The IBF flag is set and data is latched on the positive edge of $\overline{IDS}$.
2. Writing to the CSR bit 4 controls the AFLAG output and therefore the acknowledge line.

The receiver is interrupted on the negative edge of the data strobe. Data is latched in on the positive edge of the strobe pulse (see Figure 17). Since the strobe pulse is normally very short, there is little time lost between receiving the interrupt and having valid data in the input latch. The receiver is free to do other tasks prior to receiving the $\overline{INTO}$ interrupt.

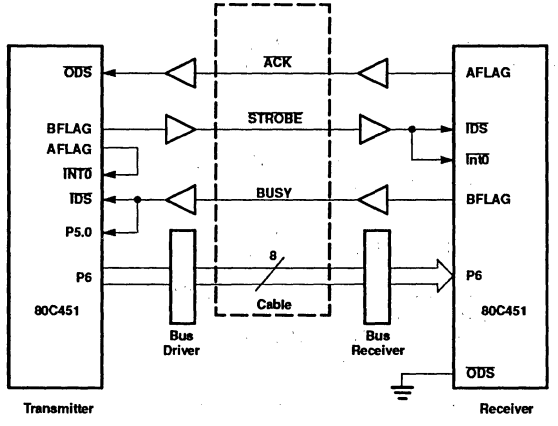


Figure 13. Interrupt Driven Parallel Interfaces Using 80C451 Controllers

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
0	1	1	0	1	1		

Figure 14. CSR Programmed for Use as an Interrupt Driven Parallel Transmitter

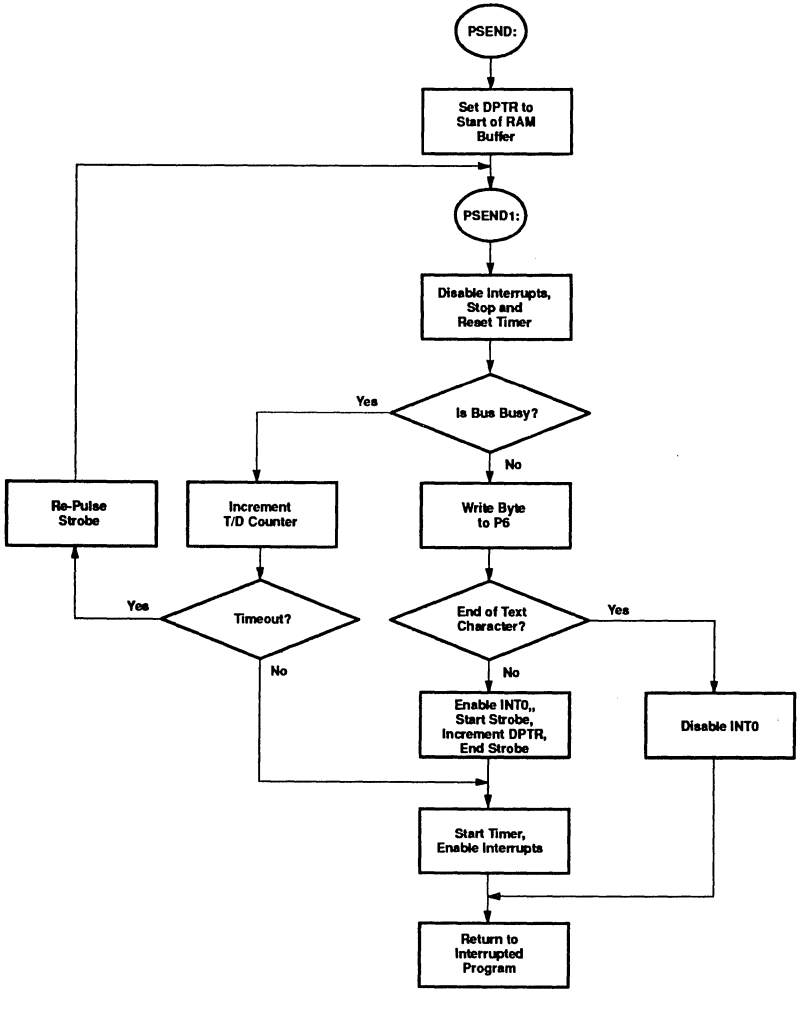


Figure 15. Flow Chart for an Interrupt Driven Parallel Transmitter

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
1	0	0	1	1	0		

Figure 16. CSR Programmed for Use as an Interrupt Driven Parallel Receiver

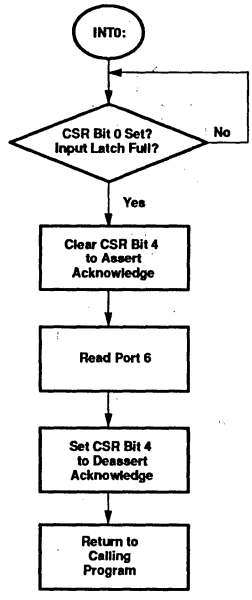


Figure 17. Flow Chart of Interrupt Driven Parallel Receiver Operation

SOFTWARE EXAMPLES

The software for the interrupt driven parallel receiver is similar to the polled receiver example. However, after an interrupt is received, this routine checks to confirm that data has been latched by the positive edge of the strobe pulse before proceeding with the routine.

```
INIT:     MOV CSR,#090H     ;INITIALIZE CSR
          SETB EX0          ;ENABLE INTERRUPT 0
          SETB IT0          ;SET NEG EDGE TRIGGERED INTERRUPTS
          SETB EA           ;ENABLE ALL INTERRUPTS
ORG EXTI0                   ;INTERRUPT 0 VECTOR
          JMP RCVR

          RCVR:
RCVR:     JNB CSR.0,#       ;CONFIRM DATA LATCHED
          CLR CSR.4         ;START ACKNOWLEDGE PULSE
          MOV R7,#02H       ;INITIALIZE THE DELAY COUNTER
          DJNZ R7,#         ;TIME ACK PULSE
          MOV A,P6          ;READ BYTE – RESET BUSY LINE
          MOV R7,#02H       ;INITIALIZE THE DELAY COUNTER
          DJNZ R7,$         ;TIME ACK PULSE
          SETB CSR.4        ;END ACK PULSE
          RET1
```

80C451 operation of port 6 AN408

This is the software for the interrupt driven parallel transmitter example.

```
; XMIT ROUTINE DRIVEN BY ACK PULSE GENERATED INTERRUPTS, OR TIME GENERATED INTERRUPTS
; FOR NON ACKNOWLEDGING PRINTERS. READS DATA BUFFER IN EXTERNAL RAM STARTING AT 100H
; AND READING UNTIL 04H IS FOUND.

        ORG   RESET
        JMP   26H
        ORG   TIMER0
        JMP   PSEND1
        ORG   EXTI0
        JMP   PSEND1
        ORG   26H
              MOV CSR,#064H          ;PORT 6 MODE
              MOV TMOD,#002H         ;CONFIGURE TIMER 0 TO 16 BITS
              SETB TO0               ;INT0 IS EDGE TRIGGERED
              SETB EA                ;ENABLE INTERRUPTS
PSEND:        MOV DPTR,#0100H        ;SET DPTR TO START OF TEXT
                                     ;BUFFER
PSEND1:       CLR EA                 ;DISABLE INTERRUPTS AND STOP
                                     ;TIMER
              CLR TR0                ;IF ENABLED
              CLR ET0
              MOV R7,00H             ;CLEAR TIMEOUT COUNTER
              MOV R6,00H
              MOV TH0,#–4            ;SET TIMER INTERRUPT PERIOD
              MOV TL0,#00H
              JB 0C8H,BB             ;BUS BUSY
              MOV ACC,#00H           ;CLEAR ACCUMULATOR
              MOVX 1,@DPTR           ;RETRIEVE FIRST BYTE
              MOV 06,ACC             ;OUTPUT FIRST BYTE
              CJNE A,#004H, CONT1    ;LOOK FOR END OF TEXT
              JMP EOTB
CONT1:        SETB ERX0              ;ENABLE INTO
              CKR 0EEH               ;START STROBE PULSE
              INC DPTR
              MOV ACC,DPH            ;LOOK FOR PHYSICAL END OF
              JB ACC.2,EOTB          ;TEXT BUFFER
              SETB 0EEH
              JMP CONT
EOTB:         CLR EX0                ;END OF TEXT FOUND, DISABLE
                                     ;INT0
              SETB 0EEH
              SETB EA
              RETI
BB:           INC R7                 ;COUNT TIMER TIMEOUTS ON
                                     ;BUS BUSY
              CJNE R7,#00H, CONT     ;LOOK FOR OVERFLOW
              INC R6                 ;COUNT OVERFLOWS
              CJNE R6,#10H, CONT     ;TIMEOUT APPROX 5 SEC
              JMP TO
CONT:         SETB TR0               ;ENABLE TIMER INTERRUPT
              SETB ET0               ;START TIMER
              SETB EA
              RETI
TO:           CLR 0C9H               ;SEND NEW STROBE PULSE IN
                                     ;RESPONSE TO TIMEOUT
              NOP
              NOP
              MOV R6,#00H            ;RESET TO COUNTER
              MOV R7,#00H
              SETB 0C9H              ;END OF STROBE PULSE
              JMP PSEND1
```

256k Centronics printer buffer using the 87C451 microcontroller

AN417

DESCRIPTION

This application note describes a stand alone Centronics type parallel printer buffer using the 87C451 expanded I/O microcontroller. This type of unit would typically be placed between a personal computer and its printer. It captures the data to be printed at high speed, freeing the personal computer to go to other tasks, and sends data to the printer as required. As described here, 256k dynamic RAMs are used, providing over one quarter million characters of storage. If desired the design is easily modified to work with 1 megabit DRAMs. Although written with the 87C451 in mind, this design is applicable to the 80C451 and 83C451.

Design Objectives

The objectives kept in mind during the design of this device were: provide a substantial size of buffer, keep the parts count and the power consumption to a minimum, and use readily available components.

A buffer size of 256k bytes was chosen because, although a 64k byte buffer is very easily implemented using the 8051 family's 64k external data storage capabilities, it is a little too small for today's printing applications that print a page of text in graphics mode, using up twenty times as many bytes as standard printing mode. Presenting a method for controlling 256k DRAMs shows off the I/O capabilities of the 87C451, and it is very easy to add the extra address line for one megabit devices if a larger buffer is needed.

The 8XC451 Microcontroller

The 8XC451 is an 8-bit microcontroller based on the familiar 8051 family of devices. In fact, it is an 80C51 with three added ports: P4, P5, and P6. Ports 4 and 5 give 12 (16 in PLCC) additional quasi-bidirectional I/O lines. Port 6 provides another 8 bits of I/O, plus 4 handshake lines that can be programmed to operate in several useful modes for interfacing. The 8XC451 comes in three versions: ROMless 80C451, 83C451 with 4k × 8 ROM, and 87C451 with 4k × 8 EPROM.

In this note, port 6 is used in the I/O mode as a Centronics compatible printer output port. Additionally, the /IDS and BFLAG pins normally associated with port 6 are used as part of the input port logic. For a complete discussion of port 6 operating modes and programming, see the application note AN408 titled "83C451 Microcontroller Operation of Port 6."

Circuit Description

Figure 1 is a schematic diagram of the printer buffer circuit. Other than the 87C451 (U1), and the eight 256k DRAMs (U5-U12), only two 74LS244 buffers (U2, U3) and a 76HCT374 (U4) octal flip-flop are needed. The U2 and U3 buffers are included to provide full drive capability for the output port and some of the handshake signals on the input port, as the output buffers on the 87C451 can only drive 3 LSTTL loads. U4 has 8-bit data strobed into it by the /STB pulse of the input port.

As the code size for this application is quite small (less than 1k bytes), the on-chip instruction memory is quite sufficient for program storage. For a production version, the 87C451 could be replaced with the 83C451 with a 4k × 8 masked ROM on chip. Note that port 0 and port 1 are not used in the present design; thus the 80C451 may be used in this application with the addition of an external address latch and EPROM.

The /RAS, /CAS, and /WR signals for the DRAM array are provided by port 3 bits /WR, /RD, and T1. Note that as in the 80C51, all port 3 signals are multifunctional. That is, each can be treated as a regular quasi-bidirectional port bit, or as having the special function indicated by its name. This feature is an advantage when using /WR and /RD as /RAS and /CAS control signals for a DRAM array. Treated as a normal port bit, the /WR pin is cleared and set by individual CLR and SETB instructions for a normal length RAM read or write cycle. However, when performing a refresh cycle, /RAS (port 3/WR) can be pulsed low using a dummy MOVX @R0,A (move to external data memory) instruction. This allows DRAM refresh to be done much more quickly than would otherwise be possible.

Port 1 and one bit from port 4 form the 9-bit address required when addressing the DRAM array. The data inputs to the array come from the parallel input data lines which are latched by U4. The RAM data outputs are fed to port 5. By making the data outputs available to the processor, it is possible to add some additional features to the firmware, such as control codes for printing multiple copies of a document, data compression, data conversion, etc. which are not implemented in this design.

Port 6 Operation

The /IDS (input data strobe) and BFLAG pins are normally used in conjunction with the port 6 bidirectional mode. In this mode, the /IDS pin is used to strobe data into the port 6 input latches, and BFLAG is used as flag output. In this application, however, these two bits are used to good effect as part of the (separate) input port logic. When a byte of data is strobed into U4 by the printer port of the host computer, the /STB signal connected to /IDS sets the input buffer full flag (IBF). BFLAG is programmed to mirror the contents of IBF, and therefore becomes asserted. This makes it ideal to be used as the BUSY output for the input port. After the input port data has been read and stored in the RAM buffer, BFLAG is de-asserted by performing a dummy read of port 6, which clears IBF. To complete the input port logic, one of the port 3 pins, P3.4, is used as the acknowledge signal, and is asserted/de-asserted by software. The /ODS pin is tied to ground to permanently enable the port 6 output drivers. This does not cause difficulty as no data is being input into the port.

Note that programming port 6 to operate in the bidirectional mode as described above means the loss of /ODS as an acknowledge input. The acknowledge input is normally used to clear the OBF (output buffer full) flag, indicating that the printer is ready for another character. On the other hand, operating port 6 in the "output only" mode causes the loss of BFLAG as BUSY output. Because the input port requires an instant BUSY indication while the output port only needs to remember the occurrence of an acknowledge pulse, it makes sense to program port 6 to operate in the bidirectional mode, with /ODS grounded to enable the output drivers. The /INT1 pin can be used instead of /ODS to record the occurrence of an acknowledge pulse with the interrupt system.

Priority and Execution of Tasks

There are three tasks that must be performed in this system: Receive—servicing the input port and storing the input character; Transmit—sending stored characters to the output port as required; and Refresh—performing DRAM refresh. The timers and interrupt system are used to manage the execution and priority of these tasks. Figure 2 and Figure 3 illustrate the flow charts of these tasks. Firmware, broken into sections, performing these three frunctions as well as an initialization routine is provided.

The 51C256 DRAMs require a 256 row refresh every 4 milliseconds. Rather than do an entire refresh cycle every 4 milliseconds, it is done as 64 rows every millisecond. This leaves time for other tasks to get service "slices" more frequently. As DRAM refresh is obviously the highest priority, timer 0 is used as the refresh interval timer, and is programmed to the 16-bit mode, and set to the higher priority level in the interrupt priority (IP) register. The refresh code is written in-line rather than in a loop to maximize speed.

An interesting point to note is that when there are no characters stored, the DRAM does not need to be refreshed. If power consumption

256k Centronics printer buffer using the 87C451 microcontroller

AN417

is of concern, the 87C451 could be programmed to go into idle mode whenever the buffer were empty. A character strobed into the input port would cause an interrupt, restarting the 87C451; DRAM refresh would be maintained until the buffer was once again empty.

The next highest priority should be input port service, as the reason for having a printer buffer is to get the data out of the computer as quickly as possible. Therefore, the input port /STB signal is connected to the /INT0 pin (as well as U4's clock pin and /IDS). Interrupt 0 is programmed in the interrupt priority register to be at the lower interrupt level so it cannot prevent refresh service. The interrupt 0 service routine stores the input character at the next location in the DRAM array, using the technique of a circular FIFO buffer. The routine also sends back an acknowledge pulse by clearing and setting the P3.4 pin, and then clears the BUSY (BFLAG) pin by performing a dummy read of port 6 (unless this character caused the buffer to be completely full).

During periods of access to the DRAM array by the input and output routines, the global interrupt enable bit (EA) is cleared so that the refresh interrupt does not disturb the contents of ports 1 and 4, or the /RAS, /CAS, and /WR signals.

The printer (output port) service routine runs all the time, except when the CPU is called to service the other conditions, therefore having the lowest priority. If there are characters in the buffer, polling is used to check for output port BUSY status. If the printer is not busy, then the character is sent, and the output port /STB pin (P4.3) is cleared and set. The output port /ACK line is connected to the /INT1 pin, so that the negative going edge of the /ACK signal is recorded as an interrupt pending. A very short INT1 service routine sets a software flag to indicate that the printer acknowledge the last character.

Possible Enhancements

There are a number of features that could be added to this design. As mentioned previously, the microcontroller could be put into the idle mode when the buffer is empty, conserving power.

The software could be enhanced to provide features such as multiple copies of a document, data compression, data conversion, automatic printer setup, etc. The PC operating system could be suitably modified to send a header for each file to be printed, containing these parameters. There is plenty of room for operating firmware expansion, and plenty of horsepower left in the 87C451 to handle these features.

The two serial port pins RxD and TxD were deliberately left unused so that input and/or output ports are easily implemented for serial interfaces or printers using the built-in UART. The pins used for parallel port handshaking could then be used as serial handshaking lines, providing the standard "modem" signals.

Combining the above two features, this circuit could act as a "splitter." By connecting a daisy-wheel printer to the serial port, a dot-matrix printer to the parallel port, and sending an "address" flag in the file header, simultaneous letter-quality and draft printing could be done.

The size of the DRAM array is easily expanded to one megabyte or large devices by connecting the additional address pins to port 4 bits 1 and 2. Only slight modifications to the operating firmware would be required.

Conclusion

The SC8XC451 microcontrollers provide plenty of I/O pins that previously had to be implemented by clumsy I/O expansion methods. The flexibility of port 6 means that this device can be used in a wide variety of applications requiring special port functions, while still using the industry standard 8051 instruction set.

The Application Note, describing a typical parallel printer buffer, makes full use of the 8XC451 features, yet allows room for enhancement and expansion.

256k Centronics printer buffer using the 87C451 microcontroller

AN417

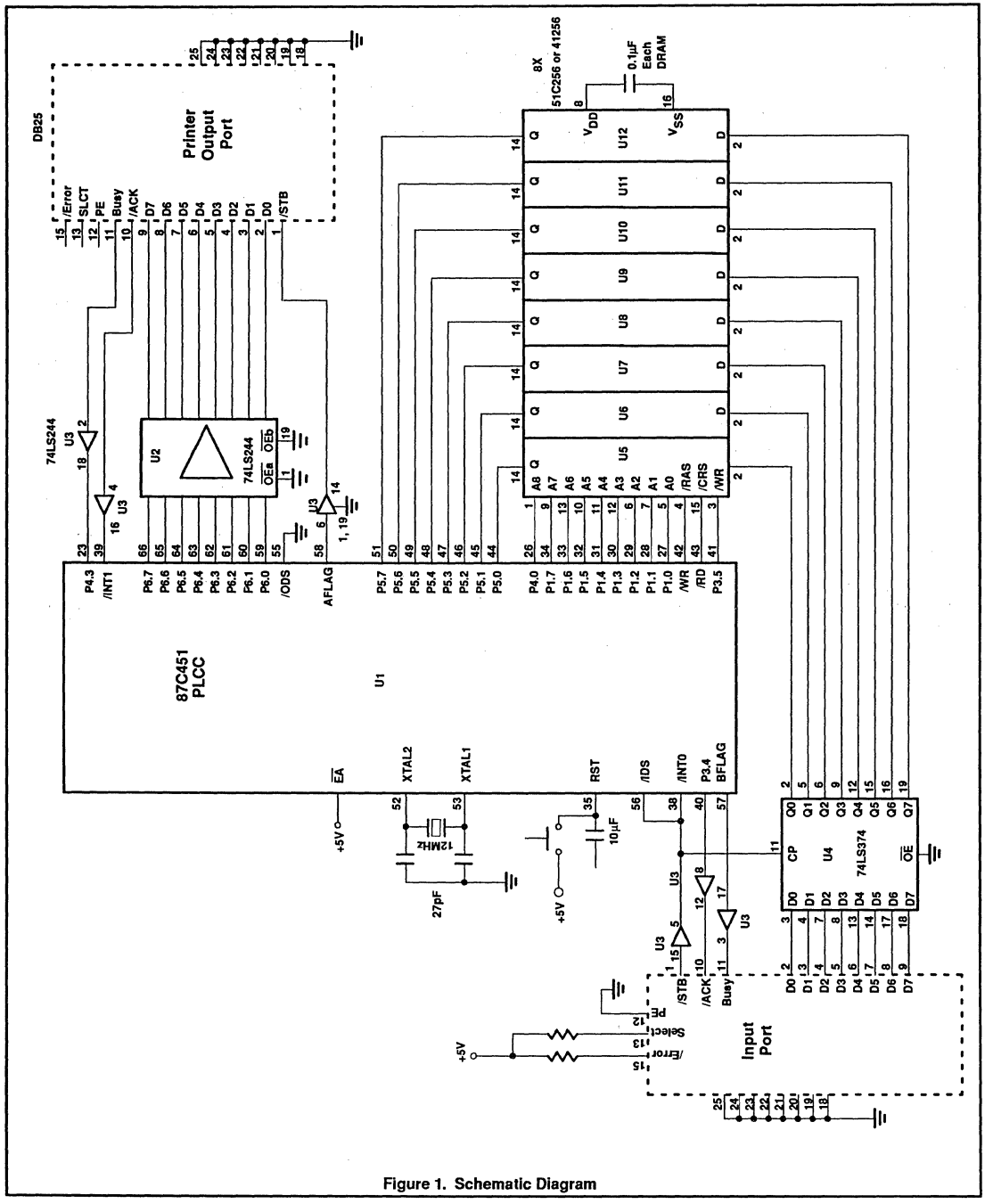


Figure 1. Schematic Diagram

256k Centronics printer buffer using the 87C451 microcontroller

AN417

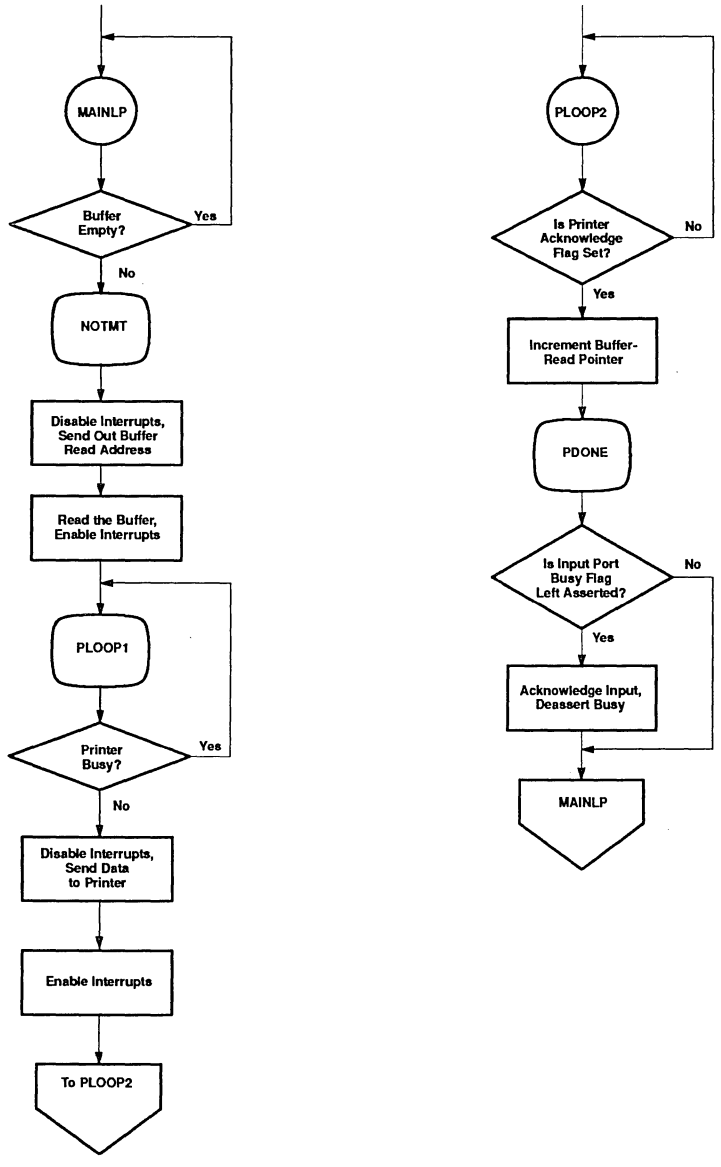


Figure 2. Flowchart of Transmit Operation

256k Centronics printer buffer using the 87C451 microcontroller

AN417

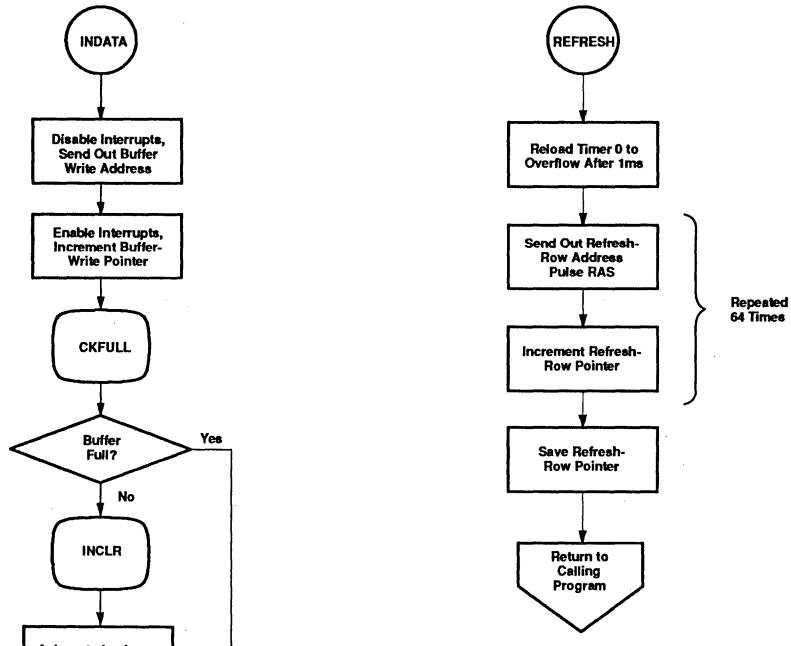


Figure 3. Flowchart of Receive and Refresh Operation

256k Centronics printer buffer using the 87C451 microcontroller

AN417

```
XMIT:       MOV DPTR,8001H
TEST:       MOVX A,@DPTR        ;READ THE CSR
            JB ACC.0,TEST       ;TEST IBF FLAG
;
;*****************************************************************************;
;                                                                             ;
;  256K PRINTER BUFFER PROGRAM USING THE 8xC451                               ;
;    FOR CENTRONICS PARALLEL PRINTER PORTS                                    ;
;                                                                             ;
;                SIGNETICS CORPORATION                                        ;
;                    OCTOBER, 1988                                            ;
;                                                                             ;
;*****************************************************************************;
;
$Mod451
$Title(*XC451 Printer Buffer)
$Date(10/28/88)
;
;  PORT USAGE:
;
;  P0          Not used (reserved for data/address bus when external
;                program memory is used).
;  P1          Lower 8 bits of DRAM address (A0 – A7).
;  P2          Not used (reserved for high-order address bus when external
;                program memory is used).
;
;  P3.0        (Reserved for serial port.)
;  P3.1        (Reserved for serial port.)
;  P3.2 (/INT0) Input port strobe input (interrupt).
;  P3.3 (/INT1) Output port acknowledge input (interrupt).
;  P3.4        Input port acknowledge output.
;  P3.5        DRAM write enable output.
;  P3.6 (/WR)  DRAM row address select output.
;  P3.7 (/RD)  DRAM column address select output.
;
;  P4.0        Upper bit of DRAM address (A8).
;  P4.1        Reserved as an extra address line for 1 megabit DRAMS.
;  P4.2        Not used.
;  P4.3        Output port busy input (OBUSY).
;  P4.4–P4.7   Unused (not available on 64-pin DIP package).
;
;  P5          DRAM output data.
;
;  P6          Parallel output port.
;  /IDS        Input port strobe input (ISTB).
;  BFLAG       Input port busy output (IBUSY).
;
;  AFLAG       Output port strobe output (OSTB).
;  /ODS        Port 6 output enable, tied low.
;
;
;  Internal Register/RAM Usage:
;
REFCNT    EQU     020h            : Low order refresh byte.;
;
;  The following refer to the circular FIFO buffer
;  implemented in the DRAM array.
;
INLOW     EQU     22h             ; Incoming address low byte.
INMID     EQU     23h             ; Incoming address mid byte.
INHI      EQU     24h             ; Incoming address high byte.
OUTLOW    EQU     25h             ; Outgoing address low byte.
OUTMID    EQU     26h             ; Outgoing address mid byte.
OUTHI     EQU     27h             ; Outgoing address high byte.
;
OACK      EQU     28h             ; Holds flag for output port acknowledge.
FOACK     BIT     OACK.0          ; Bit-address of output port acknowledge flag.
;
```

256k Centronics printer buffer using the 87C451 microcontroller


```
;   Miscellaneous Equates:
;
TIME    EQU     -1000           ; Value for 1000 timer clocks = 1 millisecond.
TIMEHI  EQU     HIGH TIME       ; High byte of timer value.
TIMELO  EQU     LOW TIME        ; Low byte of timer value.
RAS     BIT     P3.6            ; DRAM column address select.
CAS     BIT     P3.7            ; DRAM row address select.
DRAMWR  BIT     P3.5            ; DRAM write control line.
IACK    BIT     P3.4            ; Input port ACK output.
ISTB    BIT     P3.2            ; Input port strobe line (INT0).
OBUSY   BIT     P4.3            ; Output port BUSY input.
OSTB    BIT     MA0             ; Output port strobe (MA0 bit in port 6 CSR).
;
;*****************************************************************************
;
;   Reset and Interrupt Jump Table
;
            ORG     00h         ; Power-on reset.
            AJMP    START
;
            ORG     03h         ; INT 0.
            AJMP    INDATA      ; Data at input port.
;
            ORG     0Bh         ; Timer 0.
            AJMP    REFRESH     ; Refresh DRAM array.
;
            ORG     13h         ; INT 1.
            AJMP    OPACK       ; Output port acknowledge.
;*****************************************************************************
;
;   Power up reset routine:
;   Set up refresh timer, enable timer interrupt and
;   external interrupt, initialize circular buffer pointers.
;
            ORG     18h
START:      MOV     SP,#40h     ; Initialize stack pointer.
            MOV     A,#00
;           MOV     REFCNT,A    ; Initialize refresh counter.
            MOV     INLOW,A     ; Initialize FIFO pointers.
            MOV     INMID,A
            MOV     INHI,A
            MOV     OUTLOW,A
            MOV     OUTMID,A
            MOV     OUTHI,A
;
;Initialize interrupt priority register so that DRAM refresh
;   (TF0) gets high priority, input port service (IE0) and output
;   port acknowledge service get lower priority. All other
;   interrupts set to lower priority level.
;
            MOV     IE,#00000111b   ; Timer0, INT0, and INT1 enabled.
            MOV     IP,#00000010b   ; Timer0 high priority.
            MOV     TL0,#TIMELO
            MOV     TH0,#TIMEHI
            MOV     TMOD,#00000001b ; Operate Timer0 in mode 1.
            MOV     TCON,#00010101b ; Timer0 run, I0 and I1 = edge.
;
;
```

256k Centronics printer buffer using the 87C451 microcontroller


```
;  Initialize Port 6 Control and Status Register.
;  -        'BFLAG' mode set to output value of IBF
;  -         (input port BUSY signal : IBUSY)
;  -        'AFLAG' set as logic 1 output
;            (output port strobe signal : OSTB)
;  -        'IDS' active on negative level
;            (input port strobe signal : ISTB)
;            MOV     CSR,#10011100b
             MOV     A,P6            ; Dummy read of P6 to clear IBF (IBUSY).
             SETB    EA              ; Enable interrupts.
;
;*****************************************************************************
;
;  Main Routine:
;  Executes while not performing DRAM refresh or servicing
;  input port interrupt.
;
;  Check if buffer is not empty by comparing input and output
;  pointers. If not empty, go to NOTMT to output a byte.
;
MAINLP:  MOV     A,INLOW         ; Compare pointers.
         CJNE    A,OUTLOW,NOTMT
         MOV     A,INMID
         CJNE    A,OUTMID,NOTMT
         MOV     A,INHI
         CJNE    A,OUTHI,NOTMT
         SJMP    MAINLP
;
;  Buffer is not empty: compute row & column addresses for
;  a read cycle from DRAM.
;
NOTMT:   MOV     R4,OUTLOW       ; Save low byte of row.
         MOV     R5,OUTMID       ; Save upper bit of row.
         MOV     A,OUTHI         ; Shift to align correctly.
         RRC     A
         MOV     R7,A            ; Save upper column bit.
         MOV     A,OUTMID        ; Get low byte of column.
         RRC     A               ; Shift in bit from OUTHI.
         MOV     R6,A            ; Save.
;
;  Now do actual DRAM access to get the data byte at computed
;  address. Disable interrupts so we don't lose what we put
;  out on the ports.
;
         CLR     EA              ; Disable interrupts.
         MOV     P1,R4           ; Low byte row address.
         MOV     A,R5            ; Get high byte row address.
         ORL     A,#0FEh         ; Make sure OBUSY stays high.
         MOV     P4,A
         CLR     RAS             ; /RAS low.
         MOV     P1,R6           ; Low byte column address.
         MOV     A,R7            ; High byte column address.
         ORL     A,#0FEh         ; Make sure OBUSY stays high.
         MOV     P4,A
         CLR     CAS             ; /CAS low.
         MOV     R4,P5           ; Get the data byte
;
         SETB    CAS             ; /CAS high.
         SETB    RAS             ; /RAS high.
         CLR     FOACK           ; Clear acknowledge flag.
         SETB    EA              ; Re-enable interrupts.
;
```

256k Centronics printer buffer using the 87C451 microcontroller


```
PLOOP1:  JB      OBUSY,PLOOP1  ; Loop if printer busy.
;
         CLR     EA            ; Disable interrupts.
         MOV     P6,R4         ; Move byte to output port.
         CLR     MA0           ; Assert output port strobe.
         NOP                   ; Kill some time.
         NOP
         NOP
         NOP
         SETB    MA0           ; De-assert output port strobe.
         SETB    EA            ; Re-enable interrupts.
;
;  Following waits for /ACk to occur on output port. Loops on
;  acknowledge flag which is set by INT1 service routine when
;  /ACK occurs.
;
PLOOP2:  JNB     FOACK,PLOOP2  ; Wait till /ACK occurs.
;
         INC     OUTLOW        ; Increment output buffer pointer.
         MOV     A,OUTLOW
         CJNE    A,#00,PDONE
         INC     OUTMID
         MOV     A,OUTMID
         CJNE    A,#00,PDONE
         MOV     A,OUTHI
         INC     A
         ANL     A,#03h        ; Eliminate unused address bits
         MOV     OUTHI,A       ; and save.
;
;  Check if input port busy flag was left asserted, indicating that
;  the buffer was full after last input. If so, acknowledge input
;  port and de-assert input busy signal.
;
PDONE:   JNB     IBF,MAINLP    ; Not busy, return to main loop.
         CLR     EA            ; Disable interrupts.
         CLR     IACK          ; Assert /IACK.
         NOP                   ; Wait 7 microseconds.
         NOP
         NOP
         NOP
         NOP
         NOP
         NOP
         MOV     A,P6          ; Dummy read of P6 clears IBF (IBUSY).
         NOP                   ; Wait 5 microseconds.
         NOP
         NOP
         NOP
         NOP
         SETB    IACK          ; De-assert /IACK.
         SETB    EA            ; Re-enable interrupts.
         AJMP    MAINLP        ; Return to main loop.
;
```

256k Centronics printer buffer using the 87C451 microcontroller


```
;*****************************************************************************
;
;  Interrupt 1 Service Routine:
;
;  -  Called when output port asserts /ACK.
;  -  Sets FOACK flag and returns.
;
OPACK:          SETB        FOACK
        RETI
;
;*****************************************************************************
;
;  DRAM Refresh (Timer0) Interrupt Service:
;
;  -  Called once every millisecond by timer interrupt.
;  -  Refreshes 64 rows and then returns.
;  -  Therefore refreshes all rows every 4 milliseconds.
;     (Note that 41256/51C256 DRAM only requires a 256 row refresh.)
;
REFRESH:  PUSH    PSW
          MOV     TH0,#TIMEHI     ; Reload timer registers.
          MOV     TL0,#TIMELO
;
          MOV     P1,REFCNT       ; Get next row to refresh.
          MOVX    @R0,A           ; Pulse /RAS (/WR).
          INC     P1
          MOVX    @R0,A           ; 1
          INC     P1
          MOVX    @R0,A           ; 2
          INC     P1
          MOVX    @R0,A           ; 3
          INC     P1
          MOVX    @R0,A           ; 4
          INC     P1
          MOVX    @R0,A           ; 5
          INC     P1
          MOVX    @R0,A           ; 6
          INC     P1
          MOVX    @R0,A           ; 7
          INC     P1
          MOVX    @R0,A           ; 8
          INC     P1
          MOVX    @R0,A           ; 9
          INC     P1
          MOVX    @R0,A           ; 10
          INC     P1
          MOVX    @R0,A           ; 11
          INC     P1
          MOVX    @R0,A           ; 12
          INC     P1
          MOVX    @R0,A           ; 13
          INC     P1
          MOVX    @R0,A           ; 14
          INC     P1
          MOVX    @R0,A           ; 15
          INC     P1
          MOVX    @R0,A           ; 16
          INC     P1
          MOVX    @R0,A           ; 17
          INC     P1
          MOVX    @R0,A           ; 18
          INC     P1
          MOVX    @R0,A           ; 19
          INC     P1
          MOVX    @R0,A           ; 20
          INC     P1
```

```
MOVX    @R0,A       ;21
INC     P1
MOVX    @R0,A       ;22
INC     P1
MOVX    @R0,A       ;23
INC     P1
MOVX    @R0,A       ;24
INC     P1
MOVX    @R0,A       ;25
INC     P1
MOVX    @R0,A       ;26
INC     P1
MOVX    @R0,A       ;27
INC     P1
MOVX    @R0,A       ;28
INC     P1
MOVX    @R0,A       ;29
INC     P1
MOVX    @R0,A       ;30
INC     P1
MOVX    @R0,A       ;31
INC     P1
MOVX    @R0,A       ;32
INC     P1
MOVX    @R0,A       ;33
INC     P1
MOVX    @R0,A       ;34
INC     P1
MOVX    @R0,A       ;35
INC     P1
MOVX    @R0,A       ;36
INC     P1
MOVX    @R0,A       ;37
INC     P1
MOVX    @R0,A       ;38
INC     P1
MOVX    @R0,A       ;39
INC     P1
MOVX    @R0,A       ;40
INC     P1
MOVX    @R0,A       ;41
INC     P1
MOVX    @R0,A       ;42
INC     P1
MOVX    @R0,A       ;43
INC     P1
MOVX    @R0,A       ;44
INC     P1
MOVX    @R0,A       ;45
INC     P1
MOVX    @R0,A       ;46
INC     P1
MOVX    @R0,A       ;47
INC     P1
MOVX    @R0,A       ;48
INC     P1
MOVX    @R0,A       ;49
INC     P1
MOVX    @R0,A       ;50
INC     P1
MOVX    @R0,A       ;51
INC     P1
MOVX    @R0,A       ;52
INC     P1
MOVX    @R0,A       ;53
INC     P1
MOVX    @R0,A       ;54
INC     P1
```

```
            MOVX    @R0,A           ; 55
            INC     P1
            MOVX    @R0,A           ; 56
            INC     P1
            MOVX    @R0,A           ; 57
            INC     P1
            MOVX    @R0,A           ; 58
            INC     P1
            MOVX    @R0,A           ; 59
            INC     P1
            MOVX    @R0,A           ; 60
            INC     P1
            MOVX    @R0,A           ; 61
            INC     P1
            MOVX    @R0,A           ; 62
            INC     P1
            MOVX    @R0,A           ; 63
            INC     P1
;
            INC     P1              ; Adjust for next time
            MOV     REFCNT,P1       ; and save.
            POP     PSW
            RETI
;
;*****************************************************************************************
;
;   Data at Input Port:
;
;   This routine is called via interrupt INT0 whenever data
;   is strobed into the input port. It saves the data into the
;   DRAM array and increments the input pointer. If the output
;   pointer is now equal to the input pointer, then the buffer
;   is full, and we leave the busy flag set so that no more
;   data can be input until some is output and the buffer is
;   no longer full.
;
;
INDATA:     PUSH    PSW
            PUSH    ACC
            MOV     R1,INLOW        ; Lower 8 bits of row to R1.
            MOV     R2,INMID        ; Upper bit of row to R2.
            MOV     A,INHI          ; Get upper 2 bits.
            RRC     A               ; LSB to carry.
            MOV     R0,A
            MOV     A,INMID
            RRC     A               ; Shift bit into MSB.
            MOV     R3,A            ; Save.
;
            CLR     EA              ; Disable interrupts.
            MOV     P1,R1           ; LSB Row address.
            MOV     A,R2            ; MSB row address.
            ORL     A,#0FEh         ; Make sure OBUSY stays high.
            MOV     P4,A            ; MSB row address.
STBLP:      JNB     ISTB,STBLP      ; Check for end of strobe before DRAM write.
            CLR     RAS             ; /RAS low.
            CLR     DRAMWR          ; /WR low.
            MOV     P1,R3           ; LSB column address.
            MOV     1,R0            ; MSB column address.
            ORL     A,#0FEh         ; Make sure OBUSY stays high.
            MOV     P4,A            ; MSB column address.
            MOVX    A,@R0           ; Pulse /CAS low.
            SETB    RAS             ; /RAS high.
            SETB    DRAMWR          ; /WR high.
            SETB    EA              ; Re-enable interrupts.
;
```

256k Centronics printer buffer using the 87C451 microcontroller


```
            INC     INLOW           ; Increment input buffer pointer.
            MOV     A,INLOW
            CJNE    A,#00,CKFULL
            INC     INMID
            MOV     A,INMID
            CJNE    A,#00,CKFULL
            MOV     A,INHI
            INC     A
            ANL     A,#03h          ; Eliminate unused address bits.
            MOV     INHI,A
;
;   Compare input pointer to output pointer to see if the buffer is full.
;
CKFULL:     MOV     A,INLOW
            CJNE    A,OUTLOW,INCLR
            MOV     A,INMID
            CJNE    A,OUTMID,INCLR
            MOV     A,INHI
            CJNE    A,OUTHI,INCLR
;
;   If we get here, the buffer is full, so skip the acknowledge pulse.
;
            SJMP    INDONE
;
;
;   Send acknowledge pulse on /IACK line for 7 microseconds,
;   de-assert input BUSY signal halfway through.
;
INCLR:      CLR     EA              ; Disable interrupts.
            CLR     IACK            ; Assert /IACK.
            NOP                     ; Wait 7 microseconds.
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            MOV     A,P6            ; Dummy read of P6 clears IBF (IBUSY).
            NOP                     ; Wait 5 microseconds before clearing /IACK.
INDONE:     POP     ACC
            POP     PSW
            SETB    IACK            ; De-assert /IACK.
            SETB    EA              ; Re-enable interrupts.
            RETI
;
            END
```

Counter/timer 2 of the 83C552 microcontroller AN418

INTRODUCTION TO THE 83C552

The 83C552 is an 80C51 derivative with several extended features: 8k ROM, 256 bytes RAM, 10-bit A/D converter, two PWM channels, two serial I/O channels, six 8-bit I/O ports, and four counter/timers. The architecture of the 83C552 is identical to that of the 80C51, making the two devices fully code compatible. The additional peripheral functions are added to the 80C51 Special Function Register space, and the interrupt structure is modified accordingly. This information is detailed in other references on the 83C552. The focus of this application note is on one of the timers of the 83C552, Counter/Timer 2.

This counter/timer includes capture, compare, and high-speed output capabilities which facilitate many control oriented tasks. The objective of this note is to make users of the 83C552 aware of this counter/timer subsystem and assist the use of this subsystem by a detailed explanation of its operation supported by actual application examples.

TIMER 2 OF THE 83C552

Timer 2 of the 83C552 is in fact a timing controller and has an associated programmable array. The Timer 2 subsystem consists of three parts:

1. The time base consists of a 16-bit timer with a 3-bit prescaler. The master clock for the subsystem can be derived from the on-chip oscillator (fosc) or an external input, T2. It has an external reset, RT2, by which a signal applied to this input can reset the timer if the external reset is enabled.
2. A capture system consisting of four capture registers and four capture inputs which can be used for a wide variety of time measurements on external signals.
3. A compare system consisting of three compare registers and eight associated high-speed outputs which can be activated upon a match between the 16-bit timer and one of the compare registers.

For reference a complete block diagram of the 83C552 Counter/Timer 2 subsystem is shown in Figure 1.

16-BIT COUNTER/TIMER

The description of Counter/Timer 2 in the following paragraphs is intended to be a general overview. Details on architecture, address locations, interrupt structure, and timer operation are given in the 83C552 Users Manual. This users manual may be useful to complement the material presented in this application note. References to registers, bits, I/O ports, and on-chip hardware will relate directly to 83C552 Users Manual nomenclature. This application note will focus on the use of Counter/Timer 2 as a powerful input capture and high-speed output facilitator through some specific examples and not on the detailed coding.

The counter/timer consists of a 16-bit counter which is readable by software through special function registers TM2L and TM2H. The timer itself has two overflow flags, one after the entire 16-bit counter and one attached to the eighth stage. This latter flag reflects an overflow from the first byte of the counter. These two flags are present in register TM2IR and are labeled T2BO for the overflow from the first byte and T2OV for the overflow from the entire 16 bits. These flags may be used to generate an interrupt.

The counter timer is controlled directly through the special function register TM2CON, the timer 2 control register. This register also contains certain status flags.

The prescaler divides the input clock by a programmable ratio. The prescaler divide value is programmable to divide by 1, 2, 4, or 8 as controlled by T2PO and T2P1 in TM2CON.

The input clock to the prescaler is either fosc/12 or the external input, T2. The clock input to the prescaler may also be shut off. This clock input selection is controlled by bits T2MS0 and T2MS1 in TM2CON.

If T2 is used as the input clock to the timer 2 subsystem, the hardware logic samples this input and looks for a low-to-high transition. If the logic detects a logic 0 at the T2 input in state S2P1 of the microcontroller and a logic 1 in state S5P1, then this is recognized as a low-to-high transition, and the prescaler is incremented. The prescaler is incremented in the second cycle after the cycle in which the transition was detected. If the transition is detected before S2P1 is finished, the prescaler is incremented in the next cycle. This timing is shown in Figure 2. Note that this sampling rate is twice that of the normal 80C51 timers, T0 and T1; therefore T2 has twice the maximum external counting rate as compared to the standard timers.

Any programming of the clock source or the prescaler divide ratio results in a reset of the prescaler. This allows the state of the timer subsystem to be in a known state upon programming. The main 16-bit timer cannot be reset by software but it is reset by activating the reset pin or using the external reset, RT2. The external reset, RT2, can be enabled or disabled by bit T2ER in TM2CON. These resets reset the prescaler as well as the 16-bit counter.

Only one interrupt is available from the 16-bit counter timer. Two bits in TM2CON control whether TM2L, TM2H, or both flags will be used to generate the interrupt. A selection for no interrupt is also possible.

Capture System

The capture system is a powerful tool to measure the width of pulses or repetition rates. There are four independent inputs for the signals to be analyzed, CTI0 through CTI3. These inputs are alternate functions to port 1. Each input is connected to a dedicated capture register. A transition at any of these inputs will cause the content of the 16-bit counter/timer to be loaded into the respective capture register. The capture can occur upon various conditions of the input signal as specified by certain bits in the capture control register, CTCON. Each input can be set to cause a capture on a low-to-high transition, a high-to-low transition, or on both transitions. Upon a capture taking place, each input causes an interrupt flag to be set in the Timer 2 Interrupt Flag Register, TM2IR. If enabled, an interrupt will be generated.

One of the capture inputs is shown in more detail in Figure 3. All of the other capture inputs are similar to this one. The capture input is gated with the capture enable bits CTN0 and CTP0, which are located in CTCON. According to the status of these bits, the desired edges are selected to generate the capture enable pulse. The input pulse transient detection is at the input of the enable pulse generator. The input signal is sampled at S1P1 of the machine cycle. If a logic 1 is detected when a logic 0 was detected at the same time in the previous cycle, then the event is taken as a transition. An enable pulse is sent to the capture register, and the contents of timer 2 is copied into the capture register at the end of this machine cycle. The interrupt flag CTI0 is also set.

Counter/timer 2 of the 83C552 microcontroller AN418

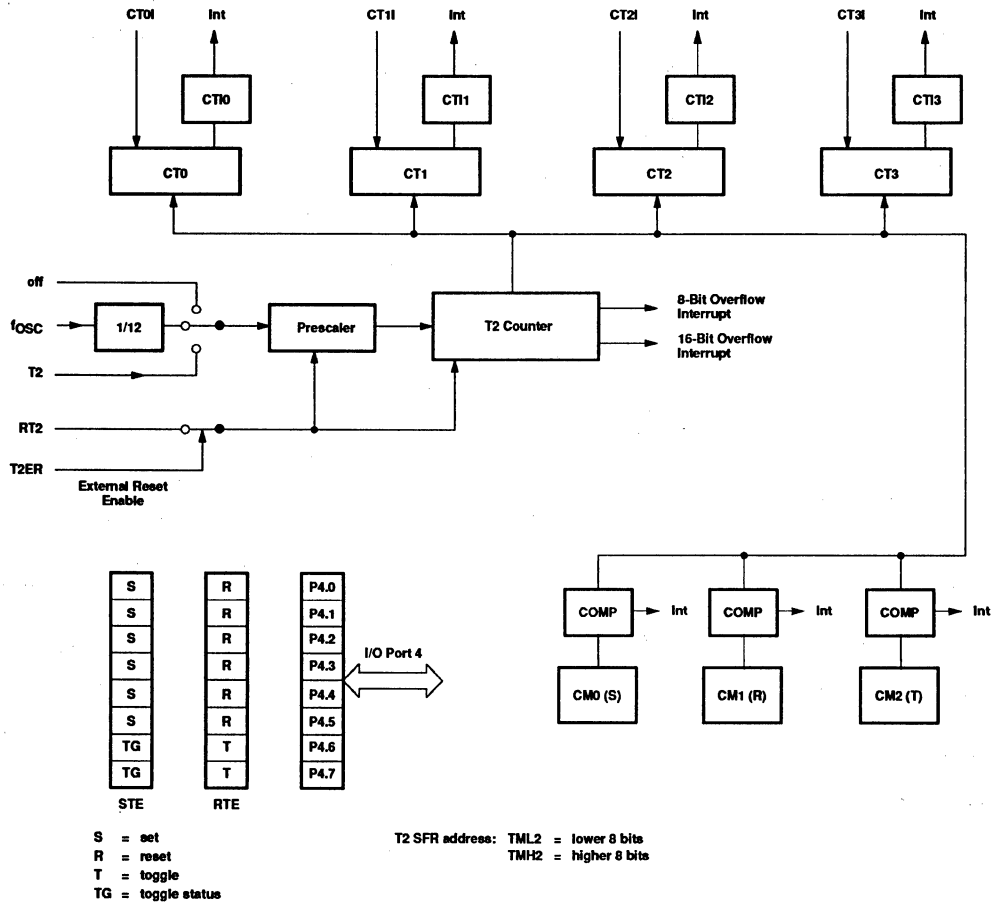


Figure 1. 83C552 Counter/Timer 2 Block Diagram

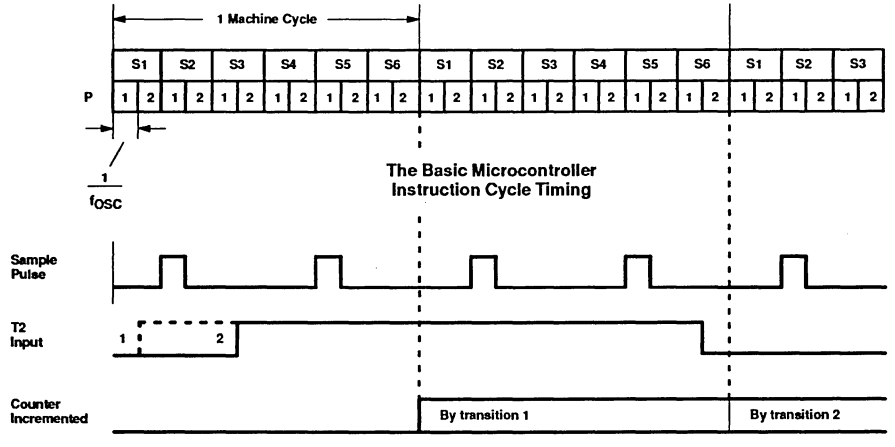


Figure 2. The Sampling of the External Clock Signal

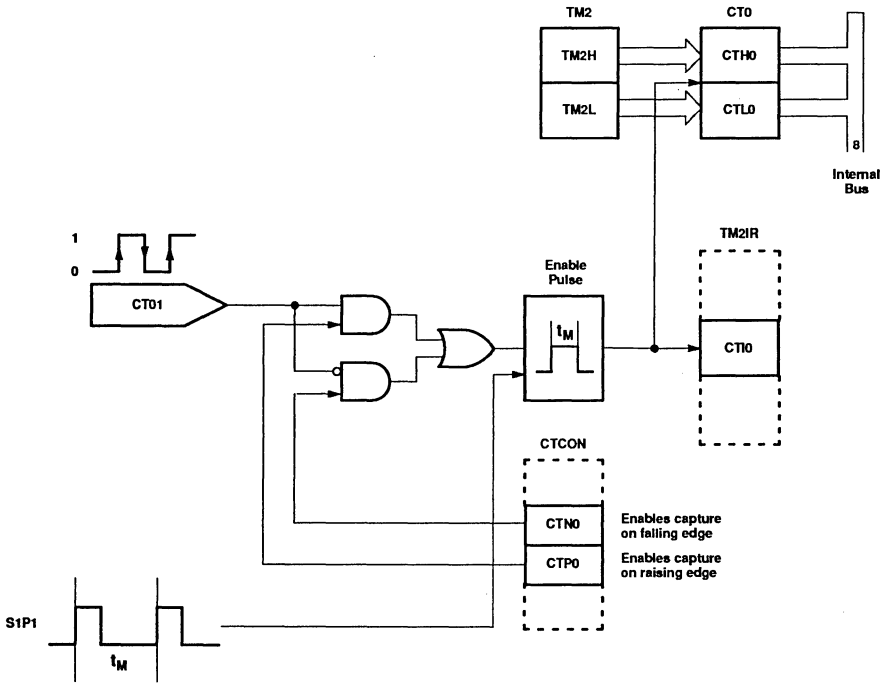


Figure 3. Capture Subsystem for CTO1

Counter/timer 2 of the 83C552 microcontroller AN418

Compare System

The compare system of Timer 2 can be used to generate a set of outputs whose transitions are controlled directly by the time defined by the 16-bit counter/timer. There are eight of these high-speed outputs which are directly controlled from Counter/Timer 2. These outputs are alternate functions to port 4. Six of these outputs are set-reset controlled (CMSR0 through CMSR5), and two are toggle controlled (CMT0 and CMT1). To clarify the operation, these two types will be discussed separately. In the following discussions, refer to Figure 4, which shows the compare system for P4.5 (a set-reset high-speed output) and P4.6 (a toggle high-speed output).

There are two compare registers associated with the set-reset outputs. These registers are CM0 and CM1. In addition, there are two enable registers: one to enable setting of an output and the other to enable resetting of an output. These registers are STE and RTE, respectively. The contents of CM0 and CM1 are continuously compared to the contents of the 16-bit counter. Whenever there is a match between the 16-bit counter and the contents of CM0, a SET pulse is generated. Similarly, whenever there is a match between the timer and the contents of CM1, a RESET pulse is generated. The set pulse is applied to the set-reset outputs, CMSR0 through CMSR5, through gates controlled by bits in STE. Bits 0 through 5 in STE control the application of the SET pulse to one of the high-speed outputs. For example, STE.0 controls the gating of the SET pulse to CMSR0, STE.1 controls the gating of the SET pulse to CMSR1, and so forth. Thus, if the corresponding SET enable bit in STE is a 1 and a compare occurs with CM0, then that high-speed output will become set. Similarly, the reset pulse from CM1 is applied to the high-speed outputs CMSR0 through CMSR5 through gates controlled by bits in RTE. As with STE, bits 0 through 5 in RTE control the application of the reset pulse to one of the high-speed outputs. If a compare occurs between the timer and CM1, a high-speed output will be reset if its corresponding enable bit in RTE is a 1. Compares with CM0 and CM1 set interrupt flags which, if enabled, can be used to generate an interrupt.

The two toggle-controlled outputs are CMT0 and CMT1, and these are associated with compare register CM2. These outputs are also alternate functions on port 4. Upon a compare between the counter and the contents of CM2, CM2 generates a toggle pulse which is applied to the high-speed outputs CMT0 and CMT1 through a set of gates. The gates control the application of the toggle pulse to the toggle outputs as specified by the high-order bits of register RTE. RET.6 controls CMT0, and RET.7 controls CMT1. Should the corresponding bit of RTE be set, then the toggle pulse is enabled to the associated high-speed output and that output will toggle upon generation of the toggle pulse from CM2. The structure of these toggled outputs is different from the other high-speed outputs in that the toggling is actually accomplished in a separate toggled flip-flop and not directly in the port latch. This toggle flip-flop cannot be controlled directly by software and powers up in an indeterminate state. The state of the toggle flip-flops is readable in STE bits 6 and 7.

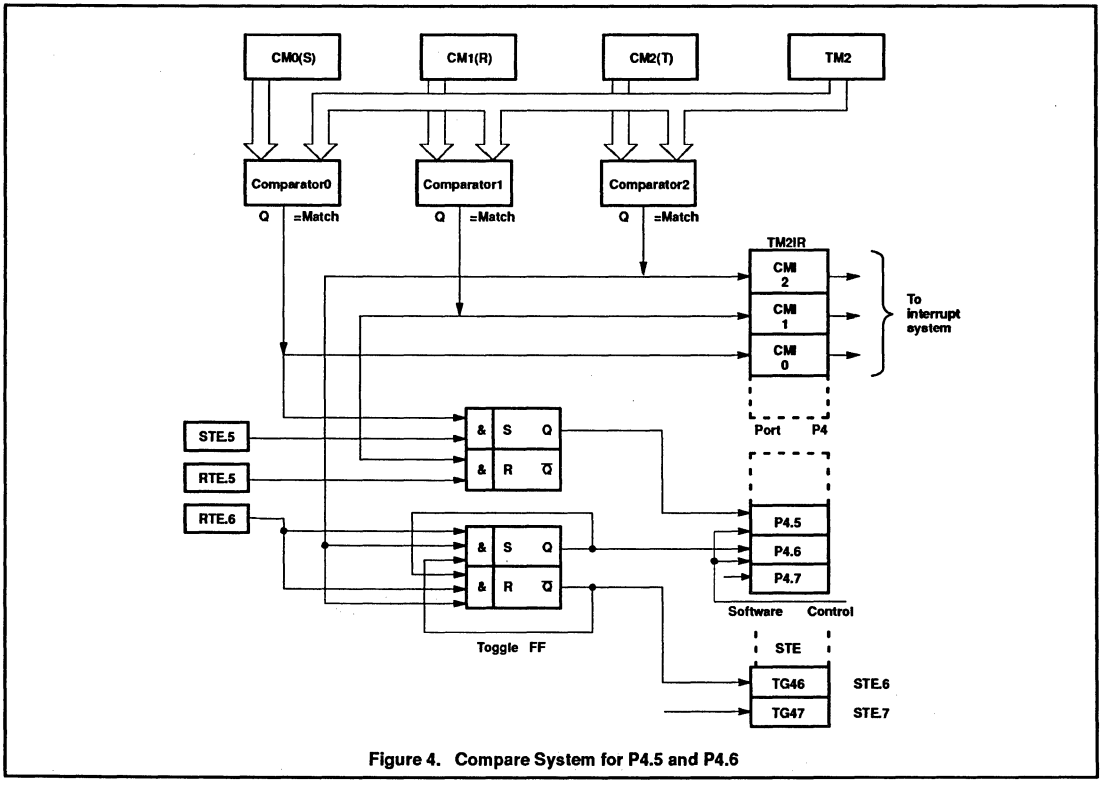


Figure 4. Compare System for P4.5 and P4.6

Counter/timer 2 of the 83C552 microcontroller AN418

APPLICATION EXAMPLE—TIMED FUEL INJECTION

In modern automobiles, optimal combustion is necessary to meet emission standards and improve fuel consumption. Optimal combustion depends on several factors and is enhanced by proper fuel injection based upon these factors which vary according to engine speed and other factors. Thus the task is to control the opening and closing of the engine fuel injectors of each cylinder relative to the crankshaft reference point.

For the application example here, we will not consider the factors which determine the timing relationships. These are assumed to be given quantities. The example here will focus upon the implementation of the injector timing control signals and how they are generated using the Counter/Timer 2 system. The illustration considers a four cylinder engine. While this is an automotive application which serves to clearly illustrate Counter/Timer 2 Subsystem operation, it is clear that many systems share similar timing requirements, and the techniques employed here are applicable to a wide class of timing tasks. The 83C552 will also support six cylinder engine control.

Figure 5 shows the injection timing required for two consecutive revolutions of the engine crankshaft.. Start and stop of the injection are given relative to a reference point on the crankshaft. The cylinders are numbered in the order of the injection sequence (not with reference to their physical location). Start of the injection is usually given in angular measure with respect to top dead center, and the injection duration is assumed to be a time value calculated from engine environmental factors and operating parameters. The angle for the start of the injection must be converted into time with respect to the reference point.

The injector drivers are assumed to be connected to the port 4 high-speed outputs CMSR0 through CMSR3. To obtain the top dead center reference point, the signal from the appropriate sensor is connected to the capture input CTOI. The interrupt for this capture input is enabled so that software can synchronize its operation to this time reference and make use of the top dead center time in the injector timing calculations. The software synchronization takes two forms. First, the captured time is an absolute reference for all real-time output operations. This time is available in capture register CT0. Note that at 12 MHz operation, Timer 2 can have a resolution as fine as 1 microsecond with a total time before overflow of over 65 milliseconds, and these times are adjustable by increasing the prescaler divide ratio. A proper selection can make the timing calculations relatively simple. Second, at the time the input is captured, flags which keep track of the phases of the crankshaft cycle are reset when cylinder 1 is at top dead center. These flags are used in the interrupt service routines to tell which action is required for that phase of the crankshaft.

Consider now the sequence of events in one rotation of the engine crankshaft and refer to Figure 5 during the discussion. Assume that the engine is running, that all relevant parameters are available, and that it has been determined that the processor is responding to the interrupt associated with the top dead center capture, CTOI. Interrupts for CTOI, CM0 (compare register 0), and CM1 (compare register 1) are enabled. Upon entering the interrupt service routine for CTOI, the previous value of the captured top dead center time is subtracted from the present value, and the crankshaft rotation time is determined. This is used to compute the time to open the first injector from the required angle at which the injector is to open. This time is made available for the interrupt service routine which responds to a compare from CM0. The interrupt service routine is exited.

The next interrupt to occur per the figure for this example is a result of a compare with CM1 which will be a result of the injector stop time for cylinder 4 having been reached. The flags in an internal status register are employed to keep track of the cylinder number that is presently active for both injection stop and injection start times. After identifying this interrupt from the flags, the processor uses the injector start time for cylinder 1 (previously loaded into CM0) and the predetermined duration to calculate the injector stop time for cylinder 1. This value is loaded into compare register CM1 and the reset enable bit for high-speed output CMSR0 is programmed to a 1. This is bit RTE.0 The reset enable bit for the cylinder 4 injector is set to 0 (bit RET.3). The interrupt routine is now exited.

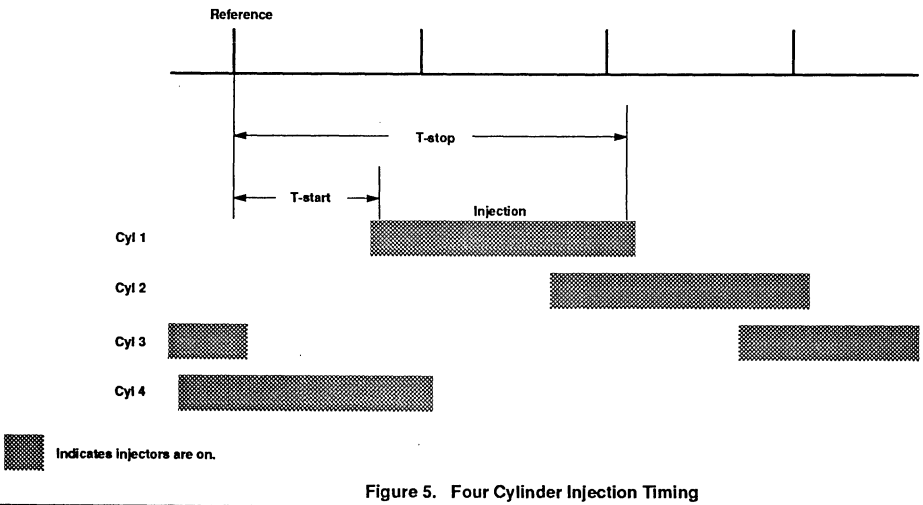


Figure 5. Four Cylinder Injection Timing

Counter/timer 2 of the 83C552 microcontroller AN418

The next interrupt to occur will be for the start time for the injector for cylinder 2. This and all subsequent cases follow the same sequence of events as for the cylinder 1 CM0 interrupt described above. In this case, calculations are made for cylinder 2 and loaded into CM0,STE.1 is programmed to a 1, and STE.0 is programmed to a 0. Similarly, the next interrupt for CM1 is treated in the same way, and the sequence of events rotates around through all cylinders in turn. The flag bits associated with this operation keep track of the injector sequencing.

While this example shows the injection stop time of one cylinder overlapping into the injector on time of the subsequent cylinder, close examination of the operations described above reveal that the start and stop events are independent and can overlap or not as required. In this way all injectors may be driven independently and have overlapping on times.

Given that this is an example applicable to general usage, it is possible that interrupt service routine could be relatively long as it would be in an actual injector application. Since the service routine has other interrupts disabled, the length may cause real-time conflicts. To eliminate this potential problem, the interrupt service routines are divided into two parts. In the first part, all other interrupts are disabled, and the essential register loading is done to prepare for the next interrupt. After this is completed, all interrupts are enabled and the ancillary service routine functions are performed prior to a return to the main routine.

As an example, consider the interrupt service routine for CM0. Upon entering the routine, all interrupts are disabled. Then the following actions are performed:

- Set bit in STE to start next injector
- Clear bit in STE for injector just started
- Load CM0 with start time for next injector
- Clear CMIO interrupt flag in TM2IR

Now that the essential set-up is made for the next interrupt, all interrupts are now enabled. However, the return to the main program is not invoked until the following ancillary processing is completed:

- Calculate the next absolute start time for the next injector (the next load value for CM0)
- Increment the flag so that the next entry to this interrupt service routine will be able to identify the next injector to start.

The process performing these calculations can be interrupted to service real-time functions.

APPLICATION EXAMPLE—TIMED IGNITION

In electronic ignition systems, multiple ignition coils may be used and each coil is fired by electronic means rather than with the old style mechanical breakers. In a four cylinder engine, there may be two ignition coils, one coil providing spark for a pair of cylinders. Both plugs fire at the same time. For one cylinder, the spark occurs at the appropriate time while for the other cylinder, the spark occurs at the end of the exhaust stroke and has no effect. With timing references to crankshaft top dead center provided by an external sensor, the ignition timing for the engine may be generated in the 83C552 and applied to the electronic drivers for the ignition coils.

To illustrate the toggle high-speed outputs of the 83C552 Counter/Timer 2 subsystem, the following example will discuss the ignition timing in a four cylinder engine employing the two coil approach with one coil for a pair of cylinders. The coil timing is illustrated in Figure 6. A reference time is used which is a given interval prior to top dead center so that the times used in the illustration can be always after the reference. There are two times of interest for each coil: the load time and the ignition point.

Ignition advance is usually given in degrees crankshaft angle prior to top dead center. As with injection, this angle is assumed to be derived from other calculations and is a given value for this illustration. This angle must be converted in to a time with respect to the reference point. The load time (the time at which the coil has to be switched on to reach the current that will give sufficient energy for an adequate spark) must be subtracted from the desired ignition point. At the ignition time, the coil will be switched off and the spark will be generated.

The coil driver electronics are connected to port bits P4.6 and P4.7. Ignition coil 1 is connected to P4.6, and ignition coil 2 is connected to P4.7. These outputs are the toggle high-speed outputs controlled by the 16-bit compare register, CM2. The program simply needs to set up the compare and control registers to turn the coils on and off at the appropriate times. It is assumed in this example that the ignition and load times are given quantities and have been determined previously.

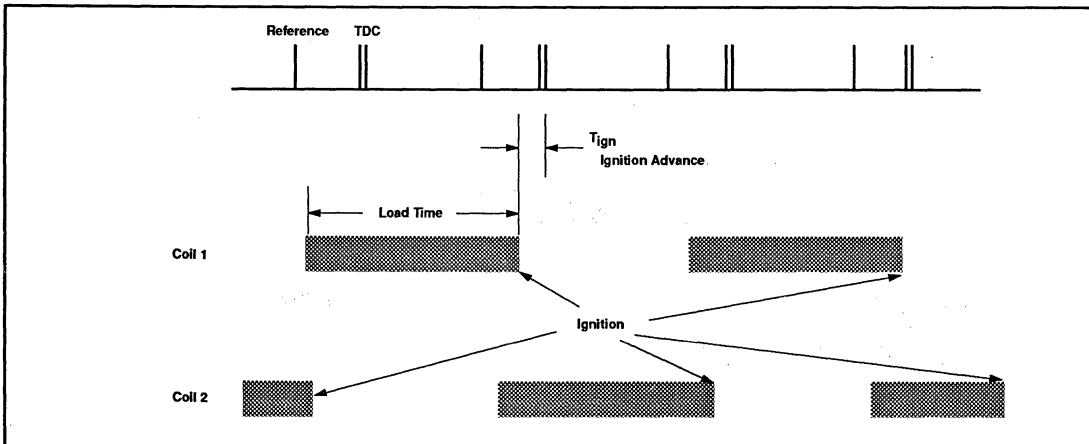


Figure 6. Four Cylinder, Two Coil Ignition Timing

Consider now the sequence of events in two rotations of the engine crankshaft and refer to Figure 6. Assume that the engine is running, that all relevant parameters are available, and that it has been determined that the processor is responding to the interrupt associated with a compare to CM2. The top dead center time and crankshaft rotation speed have been already determined through the top dead center capture, CTOI. This is the same as in the injector example. The interrupt for CTOI is enabled. From the top dead center time, the times to turn on and turn off the coil drivers are computed and made available in data storage locations in the microcontroller. It is also convenient to have flags to identify the step in the complete ignition cycle. The flags are cleared in the interrupt service routine for top dead center of cylinder 1.

Upon entering the interrupt service routine, other interrupts are disabled. Examination of the flags reveals that the state of the ignition sequence is that coil 1 has been turned on to begin the current build up (load time). The next event will therefore be turning off coil 2 to cause ignition. The interrupt service routine then performs the following actions: The time to turn off coil 2 is moved into compare register 2, CM2. Bit 6 of RTE is cleared; this disconnects the output of CM2 from the toggle flip-flop of P4.6 (coil 1). Bit 7 of RTE is set; this connects the output of CM2 to the toggle flip-flop of P4.7 (coil 2). The flags are incremented to indicate that the next interrupt will be a result of coil 2 turning off and causing ignition. The other interrupts can be enabled and a return to the main program can be executed. After the other interrupts are enabled and before a return is made to the main program, it may be convenient to do any necessary calculations to determine the time value to be loaded into CM2 in the next CM2 interrupt.

Since the flip-flops are toggled, it is likely that upon power up of the microcontroller, the toggle flip-flops will not be in the desired state. To get the toggle flip-flops in the correct state in the ignition cycle, the flip-flops must be toggled if they are in the wrong state. To determine if this is necessary, the state of the toggle flip-flops can be read from the STE register. The state of the P4.6 flip-flop is present in STE bit 6 and the state of the P4.7 flip-flop is present in STE bit 7. Comparing the actual state to the required state determines which if any or both of the flip-flops must be toggled. If a toggle is necessary to put one or both of the flip-flops in the correct state, the corresponding bits in RTE would be set for those flip-flops requiring the toggle, and CM2 would be loaded with a value that is slightly larger than the present contents of timer 2. If desired for reliability purposes, the state of the flip-flops could be checked periodically against the ignition cycle flags to determine if a correction is necessary.

CONCLUSION

This application note has examined one aspect of the 83C552 CMOS 80C51 derivative microcontroller. The Counter/Timer 2 Subsystem has been applied to a complex timing task of gasoline engine injector valve and ignition coil timing control. While this is a specific application to the automotive interests, the results are applicable to a wide variety of time measurement and control applications. The 83C552 would be ideal for many electromechanical systems such as copy machines, fax machines, industrial process control equipment, automatic transmission control, and anti-skid and anti-lock braking control.

These application areas are those which can successfully employ the 83C552 Counter/Timer 2; however, the other features should not be overlooked. When combined with the 10-bit A to D Converter, the Pulse Width Modulator, the I^2C serial bus, and peripheral device family, the 83C552 provides minimum component count solutions for cellular radio systems, professional audio systems, and medical instrumentation products such as bedside patient monitors and analyzers for home care and sports use.

Using up to 5 external interrupts on 80C51 family microcontrollers

AN420

80C51 family microcontrollers are equipped with up to two inputs which may be used as general-purpose interrupts. A typical device provides a total of 5 interrupt sources. Timer 0 and Timer 1 generate vectored interrupts, as does the Serial Port. Applications that require more than two externally signaled vectored interrupts, and do not use one or more of the counters or the serial port, can be configured to use these facilities for additional external interrupt inputs.

This note describes a method to configure the timer/counters and the serial port for use as interrupt inputs (see Figure 1). Minimum response time is a goal for this configuration.

Another popular method to implement extra interrupt inputs is to poll under software control a port pin configured as an input. This method is necessary when the on-chip peripherals are in use. Applications where this approach is recommended are ones in which the processor spends more than half of the time executing a "wait loop," or a short code sequence which jumps or branches back on itself without performing any functions. In this case, the instructions that will check the state of input used as an interrupt source are inserted into this sequence. Consequently, this input is ignored when other routines are being executed. This input may have to be latched externally, or the processor may miss the signal while executing other routines.

Dedicated interrupt inputs that vector the processor to individual service routines (as the two general-purpose interrupt inputs work) do not have the drawbacks of the method described above.

COUNTER/TIMER CONFIGURATION

Timers 0 and 1 are placed in mode 2, which configures the timer/register as an 8-bit counter with automatic reload. The counter and reload register are loaded with FF hexadecimal which is stored in TH1 and TL1 or TH0 and TL0.

To prepare one of the timers for this kind of operation, a number of control bits have to be set up. The following is a list of these bits and their values:

In TMOD:	In TCON:	In IE:
GATE = 0	TRi = 1	ETi = 1
C/T = 1		EA = 1
M1 = 1		
M0 = 0		

Where "i" is the timer number being used as the external interrupt. The TMOD value would be 66 hexadecimal if both timers are being used as external interrupt sources, x6 hex for timer 0, and 6x hex for timer 1. The interrupt priority may also be set in the IP register.

A falling edge on the corresponding Timer 0 or Timer 1 input (T0 or T1) will cause the counter to overflow and generate a timer interrupt. The counter will be automatically loaded with another FF from the reload register, so the interrupt can occur again as soon as the interrupt service routine completes. Counter/Timer operation is described in detail elsewhere in this manual.

SERIAL PORT CONFIGURATION

The serial port can be placed in mode 2, which is a 9-bit UART with the baud rate derived from the oscillator. The external interrupt is signaled through this port on the RxD receive data pin. Reception is initiated by a detected 1-to-0 transition at RxD. The signal must stay at 0 for at least five-eighths of a bit period for this level to be recognized. Refer to the description of baud rates to determine the length of a bit period at the oscillator frequency selected for the application. The input signal should remain low for at least one bit period and for not more than 9 bit periods.

To prepare the serial port for use as an external interrupt, the following bits must be set up:

In SCON:
SM0 = 1
SM1 = 0
SM2 = 0
REN = 1

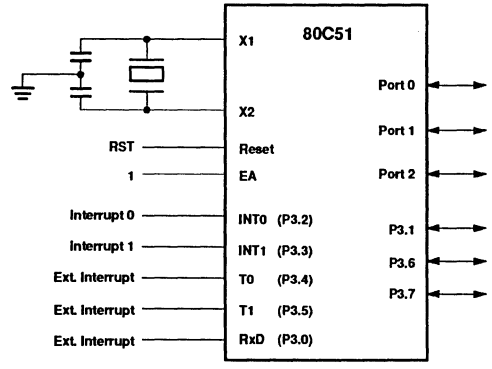


Figure 1. 80C51 Five Interrupt Configuration

Using up to 5 external interrupts on 80C51 family microcontrollers

AN420

The Serial Port Interrupt is then used as a general-purpose interrupt. The contents of receive buffer should be ignored, and will subsequently be overwritten during the next interrupt.

Note that the response time for this input will be slower than for the Counter/Timer inputs. This is due to the fact that the RI is generated after the eighth serial data bit time after the falling edge on RxD.

```
; Demonstration program for five external interrupts.

$MOD51
$TITLE (Five Vectored External Interrupts)

; Interrupt Jump Table
            ORG     OH              ;Reset
            AJMP    Setup

            ORG     3H              ;External interrupt 0.
            RETI                    ;(not implemented in this demo)

            ORG     0BH             ;Timer 0 interrupt.
            AJMP    Tim0

            ORG     13H             ;External interrupt 1.
            RETI                    ;(not implemented in this demo)

            ORG     1BH             ;Timer 1 interrupt.
            AJMP    Tim1

            ORG     23H             ;Serial port interrupt.
            AJMP    Serial

; Begin setup code
Setup       MOV     SP,#7FH         ;Initialize the stack pointer.

; Configure both timers
            MOV     TMOD,#66H       ;Put both counters into mode 2.
            MOV     A,#0FFH
            MOV     TL0,A           ;Load FF hex into both counters
            MOV     TH0,A
            MOV     TL1,A
            MOV     TH1,A
            SETB    ET0             ;Enable Timer 0 interrupt.
            SETB    ET1             ;Enable Timer 1 interrupt.
            SETB    TR0             ;Enable Timer 0 to run.
            SETB    TR1             ;Enable Timer 1 to run.

; Configure the serial port
            SETB    ES              ;Enable serial port interrupt.
            MOV     SCON,#90H       ;Put the serial port in mode 2.
            SETB    EA              ;Enable interrupt system.

Wait:       NOP                     ;Wait for an interrupt.
            JMP     Wait

Serial:     NOP                     ;Serial interrupt service routine.
            CLR     RI              ;Clear receiver interrupt flag.
            RETI

Tim0:       NOP                     ;Timer 0 interrupt service routine.
            RETI

Tim1:       NOP                     ;Timer 0 interrupt service routine.
            RETI

            END
```

Using the 8XC751 microcontroller as an I²C bus master

AN422

DESCRIPTION

The 83C751/87C751 Microcontroller offers the advantages of the 80C51 architecture in a small package and at a low cost. It combines the benefits of a high-performance microcontroller with on-board hardware supporting the Inter- Integrated Circuit (I²C) bus interface.

The I²C bus, developed and patented by Philips, allows integrated circuits to communicate directly with each other via a simple bidirectional 2-wire bus. The comprehensive family of CMOS and bipolar ICs incorporating the on-chip I²C interface offers many advantages to designers of digital control for industrial, consumer and telecommunications equipment. A typical system configuration is shown in Figure 1.

Interfacing the devices in an I²C based system is very simple because they connect directly to the two bus lines: a serial data line (SDA) and a serial clock line (SCL). System design can rapidly progress from block diagram to final schematic, as there is no need to design bus interfaces, and functional blocks on a block diagram correspond to actual ICs. A prototype system or a final product version can easily be modified or upgraded by 'clipping' or 'unclipping' ICs to or from the bus. The simplicity of designing with the I²C bus does not reduce its effectiveness; it is a reliable, multi-master bus with integrated addressing and data-transfer protocols (see Figure 2). In addition, the I²C-bus compatible ICs provide cost reduction benefits to equipment manufacturers, some of which are smaller IC packages and a minimization of PCB traces and glue logic.

The availability of microcontrollers like the 83C751, with on-board I²C interface, is a very powerful tool for system designers. The integrated protocols allow systems to be completely software defined. Software development time of different products can be reduced by assembling a library of reusable software modules. In addition, the multi-master capability allows rapid testing and alignment of end-products via external connections to an assembly-line computer.

The mask programmable 83C751 and its EPROM version, the 87C751, can operate as a master or a slave device on the I²C small area network. In addition to the efficient interface to the dedicated function ICs in the I²C family, the on-board interface facilities I/O and RAM expansion, access to EEPROM and processor-to-processor communications.

The multi-master capability of the I²C is very important but many designs do not require it. For many systems, it is sufficient that all communications between devices are initiated by a single, master processor. In this application note, use of the 8XC751 as an I²C bus master is described. Some of the technical features of the bus and the 83C751's special hardware associated with the I²C are discussed. Also included is a software example demonstrating I²C single master communications. Note that the sample routines are quite general, and therefore may be transferred easily to many applications

The discussion of the I²C bus characteristics in this application note is by no means complete. Additional information for the I²C bus and the S83C751 Microcontroller can be found in the Microcontroller Users' Guide.

THE I²C BUS

The two lines of the I²C-bus are a serial data line (SDA) and a serial clock line (SCL). Both lines are connected to a positive supply via a pull-up resistor, and remain HIGH when the bus is not busy. Each device is recognized by a unique address – whether it is a microcomputer, LCD driver, memory or keyboard interface – and can operate as either a transmitter or receiver, depending on the function of the device. A device generating a message or data is a transmitter, and a device receiving the message or data is a receiver. Obviously, a passive function like an LCD driver could only be a receiver, while a microcontroller or a memory can both transmit and receive data.

Masters and Slaves

When a data transfer takes place on the bus, a device can either be a master or a slave. The device which initiates the transfer, and generates the clock signals for this transfer, is the master. At that time any device addressed is considered a slave. It is important to note that a master could either be a transmitter or a receiver; a master microcontroller may send data to a RAM acting as a transmitter, and then interrogate the RAM for its contents acting as a receiver – in both cases performing as the master initiating the transfer. In the same manner, a slave could be both a receiver and a transmitter.

The I²C is a multi-master bus. It is possible to have, in one system, more than one device capable of initiating transfers and controlling the bus (Figure 2). A microcontroller may act as a master for one transfer, and then be the slave for another transfer, initiated by another processor on the network. The master/slave relationships on the bus are not permanent, and may change on each transfer.

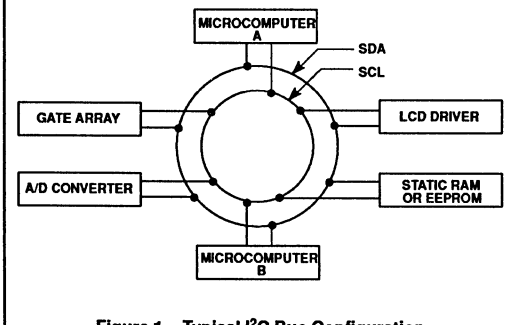


Figure 1. Typical I²C Bus Configuration

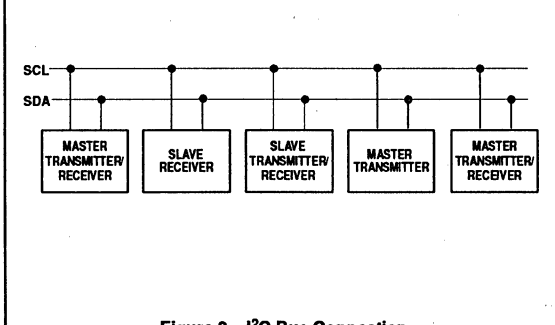


Figure 2. I²C Bus Connection

Using the 8XC751 microcontroller as an I²C bus master

AN422

As more than one master may be connected to the bus, it is possible that two devices will try to initiate a transfer at the same time. Obviously, in order to eliminate bus collisions and communications chaos, an arbitration procedure is necessary. The I²C design has an inherent arbitration and clock synchronization procedure relying on the wired-AND connection of the devices on the bus. In a typical multi-master system, a microcontroller program should allow it to gracefully switch between master and slave modes and preserve data integrity upon loss of arbitration. In this note, a simple case is presented describing the S83C751 operating as a single master on the bus.

Data Transfers

One data bit is transferred during each clock pulse (see Figure 3). The data on the SDA line must remain stable during the HIGH period of the clock pulse in order to be valid. Changes in the data line at this time will be interpreted as control signals. A HIGH-to-LOW transition of the data line (SDA) while the clock signal (SCL) is HIGH indicates a Start condition, and a LOW-to-HIGH transition of the SDA while SCL is HIGH defines a Stop condition (see Figure 4). The bus is considered to be busy after the Start condition and free again at a certain time interval after the Stop condition.

The Start and Stop conditions are always generated by the master.

The number of data bytes transferred between the Start and Stop condition from transmitter to receiver is not limited. Each byte, which must be eight bits long, is transferred serially with the most significant bit first, and is followed by an acknowledge bit. (see Figure 5). The clock pulse related to the acknowledge bit is generated by the master. The device that acknowledges has to pull down the SDA line during the acknowledge clock pulse, while the transmitting device releases the SDA line (HIGH) during this pulse (see Figure 6).

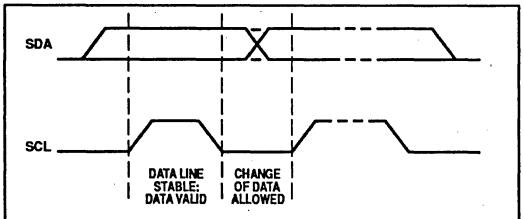


Figure 3. Bit Transfer on the I²C Bus

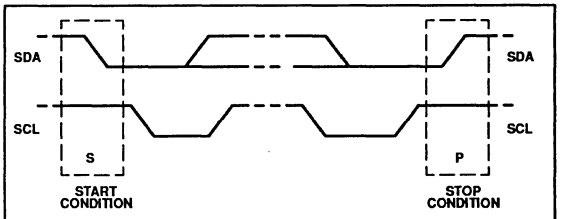


Figure 4. Start and Stop Conditions

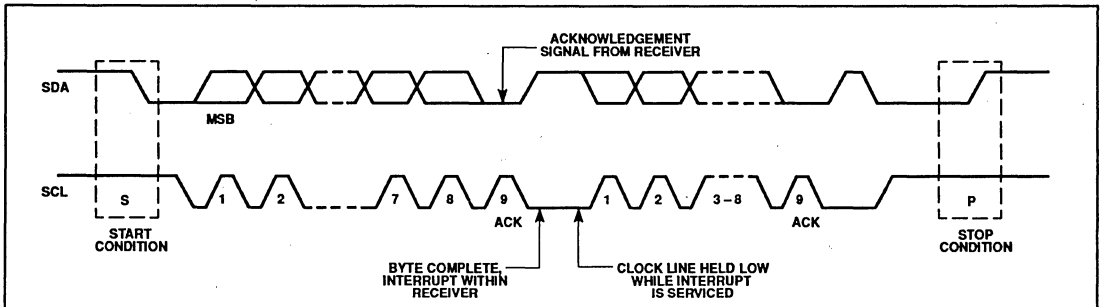


Figure 5. Data Transfer on the I²C Bus

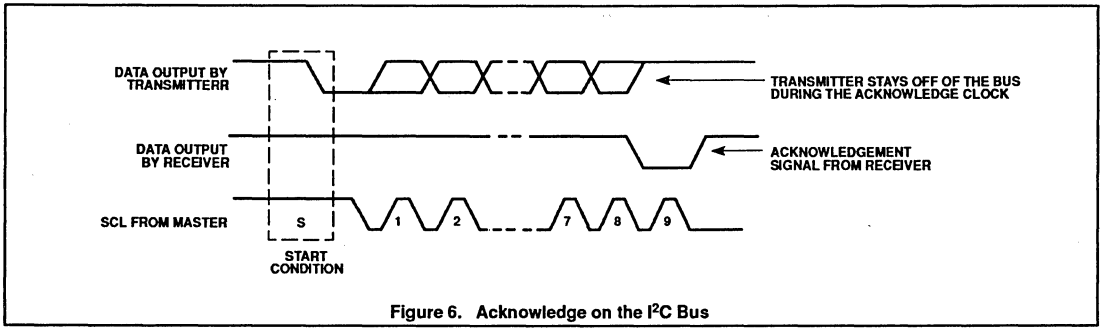


Figure 6. Acknowledge on the I²C Bus

Using the 8XC751 microcontroller as an I²C bus master

AN422

A slave receiver must generate an acknowledge after the reception of each byte, and a master must generate one after the reception of each byte clocked out of the slave transmitter. If a receiving device cannot receive the data byte immediately, it can force the transmitter into a wait state by holding the clock line (SCL) LOW. When designing a system, it is necessary to take into account cases when acknowledge is not received. This happens, for example, when the addressed device is busy in a real time operation. In such a case the master, after an appropriate "time-out", should abort the transfer by generating a Stop condition, allowing other transfers to take place. These "other transfers" could be initiated by other masters in a multi-master system, or by this same master.

There are two exceptions to the "acknowledge after every byte" rule. The first occurs when a master is a receiver: it must signal an end of data to the transmitter by NOT signalling an acknowledge on the last byte that has been clocked out of the slave. The acknowledge related clock, generated by the master should still take place, but the SDA line will not be pulled down. In order to indicate that this is an active and intentional lack of acknowledgement, we shall term this special condition as a "negative acknowledge".

The second exception is that a slave will send a negative acknowledge when it can no longer accept additional data bytes. This occurs after an attempted transfer that cannot be accepted.

The bus design includes special provisions for interfacing to microprocessors which implement all of the I²C communications in software only – it is called "Slow Mode". When all of the devices on the network have built-in I²C hardware support, the Slow Mode is irrelevant.

Addressing and Transfer Formats

Each device on the bus has its own unique address. Before any data is transmitted on the bus, the master transmits on the bus the address of the slave to be accessed for this transaction. A well-behaved slave with a matching address, if it exists on the network, should of course acknowledge the master's addressing. The addressing is done by the first byte transmitted by the master after the Start condition.

An address on the network is seven bits long, appearing as the most significant bits of the address byte. The last bit is a direction (R/W) bit. A zero indicates that the master is transmitting (WRITE) and a one indicates that the master requests data (READ). A complete data transfer, comprised of an address byte indicating a WRITE and two data bytes is shown in Figure 7.

When an address is sent, each device in the system compares the first seven bits after the Start with its own address. If there is a match, the device will consider itself addressed by the master, and will send an acknowledge. The device could also determine if in this transaction it is assigned the role of a slave receiver or slave transmitter, depending on the R/W bit.

Each node of the I²C network has a unique seven bit address. The address of a microcontroller is of course fully programmable, while peripheral devices usually have fixed and programmable address portions. In addition to the "standard" addressing discussed here, the I²C bus protocol allows for "general call" addressing and interfacing to CBUS devices.

When the master is communicating with one device only, data transfers follow the format of Figure 7, where the R/W bit could indicate either direction. After completing the transfer and issuing a Stop condition, if a master would like to address some other device on the network, it could of course start another transaction, issuing a new Start.

Another way for a master to communicate with several different devices would be by using a "repeated start". After the last byte of the transaction was transferred, including its acknowledge (or negative acknowledge), the master issues another Start, followed by address byte and data – without effecting a Stop. The master may communicate with a number of different devices, combining READS and WRITES. After the last transfer takes place, the master issues a Stop and releases the bus. Possible data formats are demonstrated in Figure 8. Note that the repeated start allows for both change of a slave and a change of direction, without releasing the bus. We shall see later on that the change of direction feature can come in handy even when dealing with a single device.

In a single master system, the repeated start mechanism may be more efficient than terminating each transfer with a Stop and starting again. In a multi-master environment, the determination of which format is more efficient could be more complicated, as when a master is using repeated starts it occupies the bus for a long time and thus preventing other devices from initiating transfers.

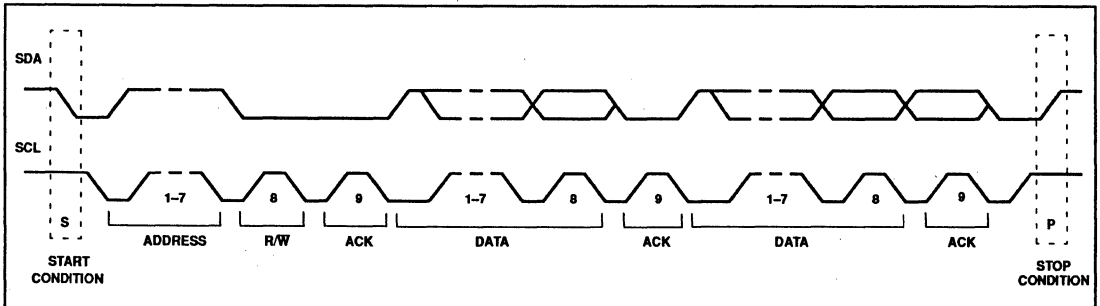


Figure 7. A Complete Data Transfer on the I²C-Bus

Using the 8XC751 microcontroller as an I²C bus master

AN422

Use of Sub-Addresses

For some ICs on the I²C bus, the device address alone is not sufficient for effective communications, and a mechanism for addressing the internals of the device is necessary. A typical example when we want to access a specific word inside the device is addressing memories, or a sequence of memory locations starting at a specific internal address.

A typical I²C memory device like the PCF8570 RAM contains a built-in word address register that is incremented automatically after each data byte which is a read or written data byte. When a master communicates with the PCF- 8570 it must send a sub-address in the byte following the slave address byte. This sub-address is the internal address of the word the master wants to access for a single byte transfer, or the beginning of a sequence of locations for a multi-byte transfer. A sub-address is an 8-bit byte, unlike the device address, it does not contain a direction (R/W) bit, and like any byte transferred on the bus it must be followed by an acknowledge.

A memory write cycle is shown in Figure 9(a). The Start is followed by a slave byte with the direction bit set to WRITE, a sub-address byte, a number of data bytes and a Stop signal. The sub-address is loaded into the word address memory, and the data bytes which follow will be written one after the other starting with the sub-address location, as the register is incremented automatically.

The memory read cycle (see Figure 9(b)) commences in a similar manner, with the master sending a slave address with the direction bit set to WRITE with a following sub-address. Then, in order to reverse the direction of the transfer, the master issues a repeated Start followed again by the memory device address, but this time with the direction bit set to READ. The data bytes starting at the internal sub-address will be clocked out of the device, each followed by a master-generated acknowledge. The last byte of the read cycle will be followed by a negative acknowledge, signalling the end of transfer. The cycle is terminated by a Stop signal.

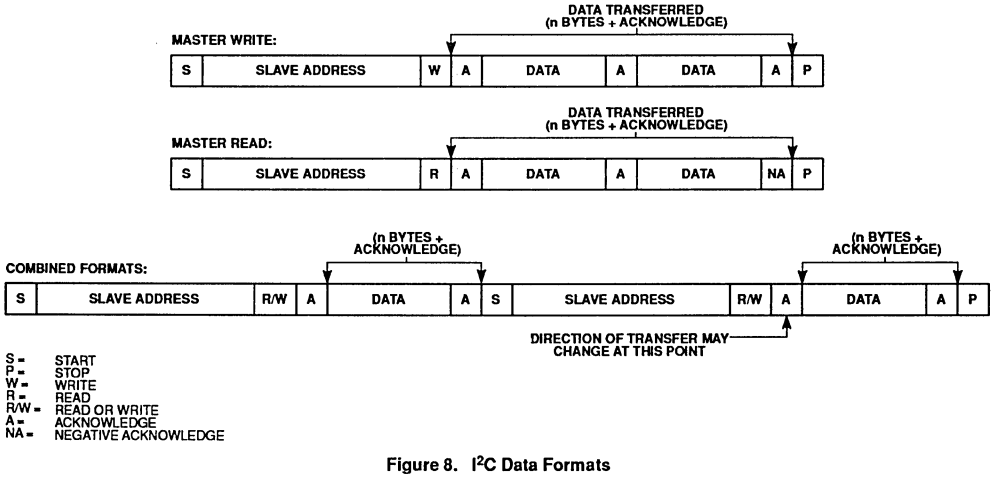


Figure 8. I²C Data Formats

Using the 8XC751 microcontroller as an I²C bus master

AN422

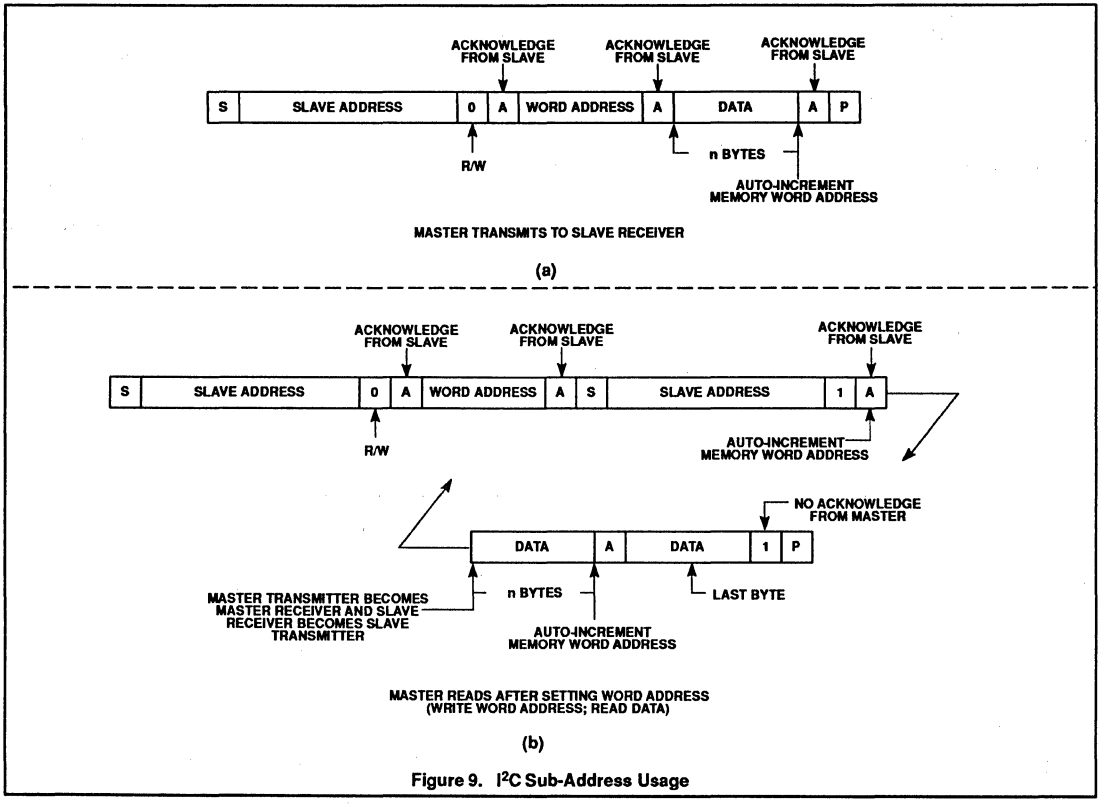


Figure 9. I²C Sub-Address Usage

Table 4. I²C Special Function Register Addresses

REGISTER			BIT ADDRESS							
Name	Symbol	Address	MSB							LSB
I²C Control	I2CON	98	9F	9E	9D	9C	9B	9A	99	98
I²C Data	I2DAT	99	–	–	–	–	–	–	–	–
I²C Configuration	I2CFG	D8	DF	DE	DD	DC	DB	DA	D9	D8

Using the 8XC751 microcontroller as an I²C bus master

AN422

8XC751 I²C HARDWARE

The on-chip I²C bus hardware support of the 8XC751 allows operation on the bus at full speed, and simplifies the software needed for effective communications on the network. The hardware activates and monitors the SDA and SCL lines, performs the necessary arbitration and framing errors checks, and takes care of clock stretching and synchronization. The hardware support includes a bus time-out timer, called Timer I. The hardware is synchronized to the software either through polled loops or interrupts.

Two of the port 0 pins are multi-functional. When the I²C is active, the pin associated with P0.0 functions as SCL, and the pin associated with P0.1 functions as SDA. These pins have an open drain output.

Two of the five 8XC751 interrupt sources may be used for I²C support. The I²C interrupt is enabled by the EI2 flag of the interrupt enable register, and its service routine should start at address 023h. An I²C interrupt is usually requested (if enabled) when a rising edge of SCL indicates a new data bit on the bus, or a special condition occurs: Start, Stop or arbitration loss. The interrupt is induced by the ATN flag – see below for the conditions for setting this flag. The Timer I overflow interrupt is enabled by the ETI flag, and the service routine starts at 01Bh.

The I²C port is controlled through three special function registers: I²C Control (I2CON), I²C Configuration (I2CFG), and I²C Data (I2DAT). The register addresses are shown in Table 4.

Although the following discussion of the hardware and register details is not complete, it should give a better understanding of the programming examples.

Timer I

In I²C applications, Timer I is dedicated to the port timing generation and bus monitoring. In non-I²C applications, it is available for use as a fixed time base.

In its port timing generation function, Timer I is used to generate SCL, the I²C clock. Timer I is clocked once per machine cycle (osc/12), so that the toggle rate of SCL will be some multiple of that rate. Because the 83C751 can be run over a wide range of oscillator frequencies, it is necessary to adjust SCL for the part's oscillator frequency. This allows the I²C bus to be used at its highest transfer rates independent of the oscillator frequency. SCL is adjusted by writing to two bits (CT0 and CT1) in the I2CFG special function register (see Table 5). The inverse of the values in CT0 and CT1 are loaded into the least significant two bit locations of Timer I every time the fourth bit of the timer is toggled. (A value is actually loaded into the least significant three bits, the third bit being 0 unless both CT0 and CT1 are programmed high and in that case the third bit is 1). SCL is then toggled every time the fourth bit of Timer I is toggled. For example: if CT1 = 0 and CT0 = 1 then the least significant three bits of Timer I would be preloaded with 2 (010 binary). Timer I would then count 3, 4, 5, 6, 7, 8 (6 counts or machine cycles). On 8, the fourth bit of Timer I will toggle, SCL will toggle and the 3 least significant bits will again be preloaded with the value 2 (010).

Table 5. CT0, CT1 Timer I Settings

CT1, CT0 Values	Timer I Counts	Oscillator Freq (MHz)
1 0	7	16
0 1	6	15, 14, 13
0 0	5	12, 11
1 1	4	10 or less

Timer I counts = f_{OSC} (MHz) x 0.39 (rounded up to next integer).

For the bus monitoring function, Timer I is used as a "watchdog timer" for bus hang-ups. It creates an interrupt when the SCL line stays in one state for an extended period of time while the bus is active (between a Start condition and a following Stop condition). SCL "stuck low" indicates a faulty master or slave. SCL "stuck high" may mean a faulty device, or that noise induced unto the I²C caused all masters to withdraw from the I²C arbitration.

The time-out interval of Timer I is fixed (cannot be set): it carries out and interrupts (if enabled) when about 1024 machine cycles have elapsed since a change on SCL within a frame. In other words, whenever I²C is active and Timer I is enabled, the falling edge of SCL will reset Timer I. If SCL is not toggled low for 1024 machine cycles, Timer I will overflow and cause an interrupt. (Note: we wrote "about 1024 machine cycles" although for the sake of accuracy – this number is affected by the setting of the CT0 and CT1 bits mentioned above and may vary by up to three machine cycles) The exact number of cycles for a time-out is not critical; what is important is that it indicates SCL is stuck.

In addition to the interrupt, upon Timer I overflow the I²C port hardware is reset. This is useful for multiple master systems in situations where a bus fault might cause the bus to hang-up due to a lack of software response. When this happens, SCL will be released, and I²C operation between other devices can continue.

I2CON Register

The I²C control register (I2CON) can be written to (see Figure 10). When writing to the I2CON register, one should use bit masks as demonstrated in the example program. Trying to clear or set the bits in the register using the bit addressing capabilities of the 8XC751 may lead to undesirable results. The reason is that a command like CLR reads the register, sets the bit and writes it back, and the write-back may affect other bits.

I2CFG Register

The configuration register (I2CFG) is a read/write register (see Figure 11).

I2DAT Register

The I²C data register (I2DAT) is a read/write register, where the MSB represents the data received or data to be sent. The other seven bits are read as 0 (see Figure 12).

Transmit Active State

The transmit active state – Xmit Active – is an internal state in the I²C interface that is affected by the I²C registers as explained above. The I²C interface will only drive the SDA line low when Xmit Active is set. Xmit Active is set by writing the I2DAT register, or by writing I2CON with XSTR = 1 or XSTP = 1. The ARL bit will be set to 1 only when Xmit Active is set – in such a case Xmit Active will be automatically reset upon arbitration loss. Xmit Active is cleared by writing 1 to CXA at I2CON register or by reading the I2DAT register.

PROGRAMMING EXAMPLE

The listing demonstrates communications routines for the 8XC751 as an I²C bus master in a single-master system.

The single-master system is less complicated than a multi-master environment. The programmer does not have to worry about switching between master and slave roles, or the consequences of an arbitration loss.

The I²C interrupt is not used, and therefore disabled. There is no need for frame Start interrupts, as this processor is the only bus master and all data transfers are initiated by it when the appropriate routines are called by the application. No one else generates frame Starts which could be an interrupt source in a multi-master system. Within the frames we monitor bus activity with a wait-loop which polls the ATN flag. As we expect the bus to operate in its full-speed mode, we can assume that only a small amount of time will

Using the 8XC751 microcontroller as an I²C bus master

AN422

be wasted in those loops, and the use of interrupts would be less efficient.

The 8XC751 has single-bit I²C hardware interface, where the registers may directly affect the levels on the bus and the software interacting with the register takes part in the protocol implementation. The hardware and the low-level routines dealing with the registers are tightly coupled. Therefore, one should take extra care if trying to modify these lower level routines.

The beginning of the program, at address 0, contains the reset vector, where the microcontroller begins executing code after a hardware reset. In this case, the code simply jumps to the main part of the program, which begins at the label 'Reset' near the end of the listing.

The main program is a simple demonstration of the I²C routines which comprise the balance of the listing. It first enables the Timer I interrupt, and sets up some sample data to be transmitted. Beginning at the label MainLoop, the program then proceeds to transmit one byte of data to a slave device at address 48 hexadecimal, using the routine titled 'SendData'. In our demonstration hardware, this address corresponds to an 8-bit I/O port that drives eight monitor LEDs. The program then reads back one byte of data from the same port using the routine 'RcvData'. The SendData and RcvData routines can send or receive multiple bytes of data, the number of which is determined by the variable 'ByteCnt'.

Upon return from both SendData and RcvData, the program checks the system flag named 'Retry' to see if the transfer was completed correctly. If not, it loops back and attempts the same transfer again.

Next, the program sends four bytes of data to a 256-byte EEPROM device, an 8-pin part called the PCF8582. The routine 'SendSub' is used for this purpose. The EEPROM was located at address A0 hexadecimal on our board. This device uses the sub-addressing feature to select a starting location to address in the EEPROM array. When data is written to the EEPROM, the address is automatically incre- mented so that the data bytes are stored in consecutive locations.

Finally the program reads back four bytes of data from the EEPROM using the routine 'RcvSub'. Calls to SendSub and RcvSub should also be followed by a test of the Retry flag to insure that all went according to plan.

This entire process is repeated indefinitely by jumping back to MainLoop.

Back at the beginning of the program, the next location after the reset vector is the Timer I interrupt service routine. The microcontroller will go to address 1B hexadecimal if Timer I overflows. This routine stops the timer, clears the timer interrupt, clears the pending interrupt so that other interrupts will be enabled, restores the stack pointer, and jumps to the 'Recover' routine to try to correct whatever stopped the I²C bus and allowed Timer I to overflow.

Next in the listing come the main I²C service routines. These are the routines SendData, RcvData, SendSub, and RcvSub that were called from the main program. Both of the send routines use the data area labeled 'XmtDat' as the transmit data buffer. In this sample program, four bytes were reserved for this area, but it could be larger or smaller depending on the application. The two receive routines use another four byte buffer labeled 'RcvDat' to store received data. All of these routines use the variables 'SlvAdr' and 'ByteCnt' to determine the slave address and the number of bytes to be sent or received, respectively. The SendSub and RcvSub routines use the variable 'SubAdr' as the sub-address to send to the slave device.

Following the main I²C service routines in the listing are the subroutines that are called by the main routines to deal intimately with the I²C hardware.

The 'SendAddr' subroutine requests mastership of the I²C bus and calls the routine 'XmitAddr' to complete sending the slave address. The bulk of the XmitAddr routine is shared with the 'XmitByte' subroutine which sends data bytes on the I²C bus. XmitByte is also used to send I²C sub-addresses. Both subroutines check for an acknowledge from the slave device after every byte is sent on the I²C bus.

The next subroutine 'RDAck' calls the 'RcvByte' routine to read in a byte of data. It then sends an acknowledge to the slave device. RDAck is used to receive all data except for the last byte of a receive data frame, where the acknowledge is omitted by the bus master. The RcvByte subroutine is called directly for the last byte of a frame.

The 'SendStop' subroutine causes a stop condition on the I²C, thus ending a frame. The 'RepStart' subroutine sends a repeated start condition on the I²C bus, to allow the master to start a new frame without first having to send an intervening stop.

The lower level subroutines deal directly with the hardware. The tight coupling between hardware and software is best demonstrated by the following explanations, relating to two cases in which the code is not self evident.

Sending the Address

When sending the address byte in the Send Addr subroutine, the first bit is written to I2DAT prior to the loop where the other seven bits are sent (SendAd2). The reason is that we need to clear the Start condition in order to release the SCL line, and this is done explicitly by the subsequent command. When SCL is released, the correct bit (MSB of address) must already be in I2DAT.

Capturing the Received Data

Typically, a program receiving data waits in a loop for ATN, and when detected, checks DRDY. If DRDY = 1 then there was a rising SCL, and the new data can be read from RDAT in I2CON or I2DAT. Reading or writing I2DAT clears DRDY, thus releasing SCL.

When reading the last bit in a byte, it should be read from I2CON, and not I2DAT (see the end of the RcvByte routine). This way the Data Ready (DRDY) flag is not cleared, and the low period on SCL is stretched. The reason for doing so is that upon reception of the last bit of a received byte the master must react with an acknowledge. In order to ensure that we "wait" with the acknowledge clock (release of SCL) until the acknowledge level is issued on SDA, the last bit is read out of I2CON and not I2DAT. SCL is stretched low until the acknowledge level is written into I2DAT by the software.

Bus Faults and Other Exceptions

Bus exceptions are detected either by Timer I time-out, or "illegal" logic states tested for and detected by the software. Upon Timer I time-out, a bus recovery is attempted by the Recover routine. The final section of the listing is this 'Recover' routine. Its job is to try to restore control of the I²C bus to the main program. First, the subroutine 'FixBus' is called. It checks to see if only the SDA line is 'stuck', and if so, tries to correct it by sending some extra clocks on the SCL line, and forcing a stop condition on the bus. If this does not work, another subroutine 'BusReset' is called. This generally happens when a severe bus error occurs, such as a shorted clock line. The philosophy used in this code is that the only chance of recovering from a severe error is to cause a reset of the I¹C hardware by deliberately forcing Timer I to time out. This method allows recovery from a temporary short or other serious condition on the I²C bus.

Using the 8XC751 microcontroller as an I²C bus master

I2CON READ

RDAT	ATN	DRDY	ARL	STR	STP	MASTER	—

RDAT Received DATa bit. The value of SDA latched by the rising edge of SCL. Its contents is identical to RDAT in the I2DAT register. Reading the received data here allows doing so without clearing DRDY and releasing SCL.

ATN An "ATteNtion" flag, set when any one of DRDY, ARL, STR or STP is set. This flag allows a single bit testing for terminating "wait loops", indicating a meaningful event on the bus. This flag also activates the I²C interrupt request.

DRDY Data ReaDY flag. Set by a rising edge of SCL when I²C is active, except at an idle slave. This flag is cleared by reading or writing the I2DAT register, or by writing a 1 to CDR (at the same address, when I2CON is written).

ARL ARbitration Loss flag. Indicates that this device lost arbitration while trying to take control of the bus.

STR STaRt flag. Set when a Start condition is detected, except at an idle slave.

STP STop flag. Set when a Stop condition is detected, except at an idle slave.

MASTER This flag is set when the controller is a bus master (or a potential master, prior to arbitration).

I2CON WRITE

CXA	IDLE	CDR	CARL	CSTR	CSTP	XSTR	XSTP

CXA "Clear Xmit Active". Writing a 1 to CXA clears the internal transmit-active state.

IDLE Setting this bit will cause a slave to enter idle mode and ignore the I²C bus until the next Start is detected. If the software sets the MASTRQ flag, the device may stop idling by turning into a master.

CDR Clear Data Ready. Clears the DRDY flag.

CARL Clear Arbitration Lost. Clears the ARL flag.

CSTR Clear STaRt. Clears the STR flag.

CSTP Clear STop. Clears the STP flag.

XSTR "Xmit repeated STaRt". Writing a 1 to this bit causes the hardware to issue a Repeated Start signal. A side effect will be setting the internal Xmit Active state. This should be used only when the device is a master.

XSTP "Xmit STop". Issues a Stop condition. The Xmit active state is set.

Figure 10. I2CON Register

SLAVEN	MASTRQ	CLRTI	TIRUN	—	—	CT1	CT0

SLAVEN Writing a 1 to this flag enables the slave functions of the I²C interface.

MASTRQ Request control of the bus as a master.

CLRTI Clear the Timer I interrupt flag. This bit is always read as 0.

TIRUN Writing a 1 will let Timer I run. When I²C is active, it will run only inside frames, and will be cleared by SCL transitions, Start and Stop. Writing a 0 will stop and clear the timer.

CT1, CT0 These bits should be programmed according to the frequency of the crystal oscillator used in the hardware. They determine the minimum high and low times for SCL, and are used to optimize performance at different oscillator speeds.

Figure 11. I2CFG Register

I2DAT READ

RDAT	—	—	—	—	—	—	—

RDAT Received DATa bit, captured from SDA every rising edge of SCL. Reading I2CAT clears DRDY and the Xmit Active state. If it is necessary to read the data without affecting the flags, it can be read out of RDAT in the I2CON register.

I2DAT WRITE

XDAT	—	—	—	—	—	—	—

XDAT Xmit DATa bit. Writing XDAT determines the data for the next bit to be transmitted on the I²C bus. Writing I2DAT also clears DRDY and sets the Xmit Active state.

Figure 12. I2DAT Register

Using the 8XC751 microcontroller as an I²C bus master

AN422

```
I2CAPP          83C751 Single Master I2C Routines                09/07/89

                 1
                 2    ;*******************************************************
                 3    ;
                 4    ;   Sample I2C Single Master Routines for the 83C751
                 5
                 6    ;*******************************************************
                 7
                 8    $TITLE(83C751 Single Master I2C Routines)
                 9    $DATE(09/07/89)
                10    $MOD751
                11    $DEBUG
                12
                13
                14    ; Value definitions.
                15
0002            16    CTVAL       EQU     02h       ;CT1, CT0 bit values for I2C.
                17
                18
                19    ; Masks for I2CFG bits.
                20
0010            21    BTIR        EQU     10h       ;Mask for TIRUN bit.
0040            22    BMRQ        EQU     40h       ;Mask for MASTRQ bit.
                23
                24
                25    ; Masks for I2CON bits.
                26
0080            27    BCXA        EQU     80h       ;Mask for CXA bit.
0040            28    BIDLE       EQU     40h       ;Mask for IDLE bit.
0020            29    BCDR        EQU     20h       ;Mask for CDR bit.
0010            30    BCARL       EQU     10h       ;Mask for CARL bit.
0008            31    BCSTR       EQU     08h       ;Mask for CSTR bit.
0004            32    BCSTP       EQU     04h       ;Mask for CSTP bit.
0002            33    BXSTR       EQU     02h       ;Mask for XSTR bit.
0001            34    BXSTP       EQU     01h       ;Mask for XSTP bit.
                35
                36
                37    ; RAM locations used by I2C routines.
                38
0021            39    BitCnt      DATA    21h      ;I2C bit counter.
0022            40    ByteCnt     DATA    22h
0023            41    SlvAdr      DATA    23h      ;Address of active slave.
0024            42    SubAdr      DATA    24h
                43
0025            44    RcvDat      DATA    25h      ;I2C receive data buffer (4 bytes).
                45                                   ; addresses 25h through 28h.
                46
0029            47    XmtDat      DATA    29h      ;I2C transmit data buffer (4 bytes).
                48                                   ;  addresses 29h through 2Ch.
                49
002D            50    StackSave   DATA    2Dh      ;Saves stack addr for bus recovery.
                51
0020            52    Flags       DATA    20h      ;I2C software status flags.
0000            53    NoAck       BIT     Flags.0  ;Indicates missing acknowledge.
0001            54    Fault       BIT     Flags.1  ;Indicates a bus fault of some kind.
0002            55    Retry       BIT     Flags.2  ;Indicates that last I2C transmission
                56                                   ; failed and should be repeated.
                57
0080            58    SCL         BIT     P0.0     ;Port bit for I2C serial clock line.
0081            59    SDA         BIT     P0.1      ;Port bit for I2C serial data line.
                60
```

```
                61   ;*************************************************************
                62   ;                          Begin Code
                63   ;*************************************************************
                64
                65   ; Reset and interrupt vectors.
                66
0000 21E1       67            AJMP    Reset             ;Reset vector at address 0.
                68
                69
                70   ; A timer I timeout usually indicates a 'hung' bus.
                71
001B            72            ORG     1Bh               ;Timer I (I2C timeout)
                                                        ; interrupt.
001B D2DD       73   TimerI:   SETB    CLRTI             ;Clear timer I interrupt.
001D C2DC       74            CLR     TIRUN
001F 1126       75            ACALL   ClrInt            ;Clear interrupt pending.
0021 852D81     76            MOV     SP,StackSave      ;Restore stack for return
                                                        ; to main.
0024 218A       77            AJMP    Recover           ;Attempt bus recovery.
0026 32         78   ClrInt:  RETI
                79
                80
                81   ;*************************************************************
                82   ;              Main Transmit and Receive Routines
                83   ;*************************************************************
                84
                85   ; Send data byte(s) to slave.
                86   ;  Enter with slave address in SlvAdr, data in XmtDat buffer,
                87   ;   # of data bytes to send in ByteCnt.
                88
0027 C200       89   SendData: CLR     NoAck             ;Clear error flags.
0029 C201       90            CLR     Fault
002B C202       91            CLR     Retry
002D 85812D     92            MOV     StackSave,SP      ;Save stack address
                                                          for bus fault.
0030 E523       93            MOV     A,SlvAdr          ;Get slave address.
0032 310C       94            ACALL   SendAddr          ;Get bus and send slave addr.
0034 200012     95            JB      NoAck,SDEX        ;Check for missing
                                                        ; acknowledge.
0037 200112     96            JB      Fault,SDatErr     ;Check for bus fault.
003A 7829       97            MOV     R0,#XmtDat        ;Set start of transmit
                                                        ; buffer.
                98
003C E6         99   SDLoop:  MOV     A,@R0             ;Get data for slave.
003D 08         100           INC     R0
003E 3125       101           ACALL   XmitByte          ;Send data to slave.
0040 200006     102           JB      NoAck,SDEX        ;Check for missing
                                                        ; acknowledge.
0043 200106     103           JB      Fault,SDatErr     ;Check for bus fault.
0046 D522F3     104           DJNZ    ByteCnt,SDLoop
                105
0049 3166       106   SDEX:   ACALL   SendStop          ;Send an I2C stop.
004B 22         107           RET
                108
                109
                110  ; Handle a transmit bus fault.
                111
004C 218A       112  SDatErr:  AJMP    Recover          ;Attempt bus recovery.
                113
                114
                115     ; Receive data byte(s) from slave.
                116     ; Enter with slave address in SlvAdr,
                        ;  # of data bytes requested in ByteCnt.
                117     ;  Data returned in RcvDat buffer.
```

Using the 8XC751 microcontroller as an I²C bus master

AN422

```
                    118
004E C200           119   RcvData:  CLR     NoAck             ;Clear error flags.
0050 C201           120             CLR     Fault
0052 C202           121             CLR     Retry
0054 85812D         122             MOV     StackSave,SP      ;Save stack address
                                                              ; for bus fault.
0057 E523           123             MOV     A,SlvAdr          ;Get slave address.
0059 D2E0           124             SETB    ACC.0             ;Aet bus read bit.
005B 310C           125             ACALL   SendAddr          ;Send slave address.
005D 200023         126             JB      NoAck,RDEX        ;Check for missing
                                                              ; acknowledge.
0060 200123         127             JB      Fault,RDatErr     ;Check for bus fault.
                    128
0063 7825           129             MOV     R0,#RcvDat        ;Set start of receive
                                                              ; buffer.
0065 D52202         130             DJNZ    ByteCnt,RDLoop    ;Check for count = 1
                                                              ; byte only.
0068 800A           131             SJMP    RDLast
                    132
006A 3143           133   RDLoop:   ACALL   RDAck             ;Get data and send
                                                              ; an acknowledge.
006C 200117         134             JB      Fault,RDatErr     ;Check for bus fault.
006F F6             135             MOV     @R0,A             ;Save data.
0070 08             136             INC     R0
0071 D522F6         137             DJNZ    ByteCnt,RDLoop    ;Repeat until last
                                                              ; byte.
                    138
0074 314F           139   RDLast:   ACALL   RcvByte           ;Get last data byte
                                                              ; from slave.
0076 20010D         140             JB      Fault,RDatErr     ;Check for bus
                                                              ; fault.
0079 F6             141             MOV     @R0,A             ;Save data.
                    142
007A 759980         143             MOV     I2DAT,#80h        ;Send negative
                                                              ; acknowledge.
007D 309EFD         144             JNB     ATN,$             ;Wait for NAK sent.
0080 309D03         145             JNB     DRDY,RDatErr      ;Check for bus
                                                              ; fault.
                    146
0083 3166           147   RDEX:     ACALL   SendStop          ;Send an I2C bus
                                                              ; stop.
0085 22             148             RET
                    149
                    150
                    151   ; Handle a receive bus fault.
                    152
0086 218A           153   RDatErr:  AJMP    Recover           ;Attempt bus recovery.
                    154
                    155
                    156   ; Send data byte(s) to slave with subaddress.
                    157   ; Enter with slave address in ACC, subaddress in
                          ; SubAdr, # of bytes to send in ByteCnt,
                    158   ; data in XmtDat buffer.
                    159
0088 C200           160   SendSub:  CLR     NoAck             ;Clear error flags.
008A C201           161             CLR     Fault
008C C202           162             CLR     Retry
008E 85812D         163             MOV     StackSave,SP      ;Save stack address
                                                              ; for bus fault.
0091 E523           164             MOV     A,SlvAdr          ;Get slave address.
0093 310C           165             ACALL   SendAddr          ;Get bus and send
                                                              ; slave address.
```

Using the 8XC751 microcontroller as an I²C bus master

AN422

```
0095 20001C   166              JB      NoAck,SSEX      ;Check for missing
                                                       ; acknowledge.
0098 20011C   167              JB      Fault,SSubErr   ; Check for bus
                                                       ; fault.
              168
009B E524     169              MOV     A,SubAdr        ;Get slave subaddress.
009D 3125     170              ACALL   XmitByte        ;Send subaddress.
009F 200012   171              JB      NoAck,SSEX      ;Check for missing
                                                       ; acknowledge.
00A2 200112   172              JB      Fault,SSubErr   ;Check for bus fault.
00A5 7829     173              MOV     R0,#XmtDat      ;Set start of
                                                       ; transmit buffer.
              174
00A7 E6       175   SSLoop:    MOV     A,@R0           ;Get data for slave.
00A8 08       176              INC     R0
00A9 3125     177              ACALL   XmitByte        ;Send data to slave.
00AB 200006   178              JB      NoAck,SSEX      ;Check for missing
                                                       ; acknowledge.
00AE 200106   179              JB      Fault,SSubErr   ;Check for bus fault.
00B1 D522F3   180              DJNZ    ByteCnt,SSLoop
              181
00B4 3166     182   SSEX:      ACALL   SendStop        ;Send an I2C stop.
00B6 22       183              RET
              184
              185
              186   ; Handle a transmit bus fault.
              187
00B7 218A     188   SSubErr:   AJMP    Recover         ;Attempt bus recovery.
              189
              190
              191   ; Receive data byte(s) from slave with subaddress.
              192   ; Enter with slave address in SlvAdr, subaddress in SubAdr,
                    ; # of data bytes  requested in ByteCnt.
              193   ; Data returned in RcvDat buffer.
              194
00B9 C200     195   RcvSub:    CLR     NoAck           ;Clear error flags.
00BB C201     196              CLR     Fault
00BD C202     197              CLR     Retry
00BF 85812D   198              MOV     StackSave,SP    ;Save stack address
                                                       ; for bus fault.
00C2 E523     199              MOV     A,SlvAdr        ;Get slave address.
00C4 310C     200              ACALL   SendAddr        ;Send slave address.
00C6 20003E   201              JB      NoAck,RSEX      ;Check for missing
                                                       ; acknowledge.
00C9 20013E   202              JB      Fault,RSubErr   ;Check for bus fault.
              203
00CC E524     204              MOV     A,SubAdr        ;Get slave subaddress.
00CE 3125     205              ACALL   XmitByte        ;Send subaddress.
00D0 200034   206              JB      NoAck,RSEX      ;Check for missing
                                                       ; acknowledge.
00D3 200134   207              JB      Fault,RSubErr   ;Check for bus fault.
              208
00D6 317A     209              ACALL   RepStart        ;Send repeated start.
00D8 20012F   210              JB      Fault,RSubErr   ;Check for bus fault.
00DB E523     211              MOV     A,SlvAdr        ;Get slave address.
00DD D2E0     212              SETB    ACC.0           ;Set bus read bit.
00DF 3115     213              ACALL   SendAd2         ;Send slave address.
00E1 200023   214              JB      NoAck,RSEX      ;Check for missing
                                                       ; acknowledge.
00E4 200123   215              JB      Fault,RSubErr   ;Check for bus fault.
              216
00E7 7825     217              MOV     R0,#RcvDat      ;Set start of
                                                       ; receive buffer.
00E9 D52202   218              DJNZ    ByteCnt,RSLoop  ;Check for count = 1
                                                       ; byte only.
```

```
00EC 800A          219              SJMP    RSLast
                   220
00EE 3143          221   RSLoop:    ACALL   RDAck           ;Get data and send
                                                             ; an acknowledge.
00F0 200117        222              JB      Fault,RSubErr   ;Check for bus fault.
00F3 F6            223              MOV     @R0,A           ;Save data.
00F4 08            224              INC     R0
00F5 D522F6        225              DJNZ    ByteCnt,RSLoop  ;Repeat until last byte.
                   226
00F8 314F          227   RSLast:    ACALL   RcvByte         ;Get last data byte
                                                             ; from slave.
00FA 20010D        228              JB      Fault,RSubErr   ;Check for bus fault.
00FD F6            229              MOV     @R0,A           ;Save data.
                   230
00FE 759980        231              MOV     I2DAT,#80h      ;Send negative
                                                             ; acknowledge.
0101 309EFD        232              JNB     ATN,$           ;Wait for NAK sent.
0104 309D03        233              JNB     DRDY,RSubErr    ;Check for bus fault.
                   234
0107 3166          235   RSEX:      ACALL   SendStop        ;Send an I2C bus stop.
0109 22            236              RET
                   237
                   238
                   239   ; Handle a receive bus fault.
                   240
010A 218A          241   RSubErr:   AJMP    Recover         ;Attempt bus recovery.
                   242
                   243
                   244   ;*********************************************************
                   245   ;                    Subroutines
                   246   ;*********************************************************
                   247
                   248   ; Send address byte.
                   249   ;  Enter with address in ACC.
                   250
010C 75D852        251   SendAddr:  MOV     I2CFG,#BMRQ+BTIR+CTVAL   ;Request I2C bus.
010F 309EFD        252              JNB     ATN,$                    ;Wait for bus
                                                                     ; granted.
0112 309908        253              JNB     Master,SAErr             ;Should have
                                                                     ; become the bus
                                                                     ; master.
0115 F599          254   SendAd2:   MOV     I2DAT,A                  ;Send first bit,
                                                                     ; clears DRDY.
0117 75981C        255              MOV     I2CON,#BCARL+BCSTR+BCSTP ;Clear start,
                                                                     ; releases SCL.
011A 3120          256              ACALL   XmitAddr                 ;Finish sending
                                                                     ; address.


011C 22            257              RET
                   258
011D D201          259   SAErr:     SETB    Fault                    ;Return bus fault
                                                                     ; status.
011F 22            260              RET
                   261
                   262
                   263   ; Byte transmit routine.
                   264   ;   Enter with data in ACC.
                   265   ;   XmitByte : transmits 8 bits.
                   266   ;   XmitAddr : transmits 7 bits (for address only).
                   267
0120 752108        268   XmitAddr:  MOV     BitCnt,#8            ;Set 7 bits of
                                                                  ; address count.
0123 8005          269              SJMP    XmBit2
                   270
```

```
0125 752108      271   XmitByte:  MOV     BitCnt,#8          ;Set 8 bits of data
                                                             ; count.
0128 F599        272   XmBit:     MOV     I2DAT,A            ;Send this bit.
012A 23          273   XmBit2:    RL      A                  ;Get next bit.
012B 309EFD      274              JNB     ATN,$              ;Wait for bit sent.
012E 309D0F      275              JNB     DRDY,XMErr         ;Should be data ready.
0131 D521F4      276              DJNZ    BitCnt,XmBit       ;Repeat until all bits sent.
0134 7598A0      277              MOV     I2CON,#BCDR+BCXA   ;Switch to
                                                             ; receive mode.
0137 309EFD      278              JNB     ATN,$              ;Wait for acknowledge
                                                             ; bit.
013A 309F02      279              JNB     RDAT,XMBX          ;Was there an ack?
013D D200        280              SETB    NoAck              ;Return no acknowledge
                                                             ; status.
013F 22          281   XMBX:      RET
                 282
0140 D201        283   XMErr:     SETB    Fault              ;Return bus fault
                                                             ; status.
0142 22          284              RET
                 285
                 286
                 287   ; Byte receive routines.
                 288   ;   RDAck    : receives a byte of data, then sends
                       ;   an acknowledge.
                 289   ;   RcvByte : receives a byte of data.
                 290   ;   Data returned in ACC.
                 291
0143 314F        292   RDAck:     ACALL   RcvByte            ;Receive a data byte.
0145 759900      293              MOV     I2DAT,#0           ;Send receive
                                                             ; acknowledge.
0148 309EFD      294              JNB     ATN,$              ;Wait for acknowledge
                                                             ; sent.
014B 309D15      295              JNB     DRDY,RdErr         ;Check for bus fault.
014E 22          296              RET
                 297
014F 752108      298   RcvByte:   MOV     BitCnt,#8          ;Set bit count.
0152 E4          299              CLR     A                  ;Init received byte
                                                             ; to 0.
0153 4599        300   RBit:      ORL     A,I2DAT            ;Get bit, clear ATN.
0155 23          301              RL      A                  ;Shift data.
0156 309EFD      302              JNB     ATN,$              ;Wait for next bit.
0159 309D07      303              JNB     DRDY,RdErr         ;Should be data ready.
015C D521F4      304              DJNZ    BitCnt,RBit        ;Repeat until 7 bits
                                                             ; are in.
015F A29F        305              MOV     C,RDAT             ;Get last bit, don't
                                                             ; clear ATN.
0161 33          306              RLC     A                  ;Form full data byte.
0162 22          307              RET
                 308
0163 D201        309   RdErr:     SETB    Fault              ;Return bus fault status.
0165 22          310              RET
                 311
                 312
                 313   ; I2C stop routine.
                 314
0166 C2DE        315   SendStop:  CLR     MASTRQ             ;Release bus
                                                             ; mastership.
0168 759821      316              MOV     I2CON,#BCDR+BXSTP  ;Generate a bus stop.
016B 309EFD      317              JNB     ATN,$              ;Wait for atn.
016E 759820      318              MOV     I2CON,#BCDR        ;Clear data ready.
0171 309EFD      319              JNB     ATN,$              ;Wait for stop sent.
0174 759894      320              MOV     I2CON,#BCARL+BCSTP+BCXA ;Clear I2C bus.
0177 C2DC        321              CLR     TIRUN              ;Stop timer I.
0179 22          322              RET
                 323
```

Using the 8XC751 microcontroller as an I²C bus master

AN422

```
                    324
                    325     ; I2C repeated start routine.
                    326     ;   Enter with address in ACC.
                    327
017A 759822         328     RepStart:  MOV     I2CON,#BCDR+BXSTR   ;Send repeated start.
017D 309EFD         329                JNB     ATN,$               ;Wait for ATN.
0180 759820         330                MOV     I2CON,#BCDR         ;Clear data ready.
0183 309EFD         331                JNB     ATN,$               ;Wait for repeated
                                                                   ; start sent.
0186 759818         332                MOV     I2CON,#BCARL+BCSTR  ;Clear start.
0189 22             333                RET
                    334
                    335
                    336     ; Bus fault recovery routine.
                    337
018A 31A4           338     Recover:   ACALL   FixBus              ;See if bus is dead or
                                                                   ; can be 'fixed'.
018C 400D           339                JC      BusReset            ;If not 'fixed', try
                                                                   ; extreme measures.
018E D202           340                SETB    Retry               ;If bus OK, return to
                                                                   ; main routine.
0190 C201           341                CLR     Fault
0192 C200           342                CLR     NoAck
0194 D2DD           343                SETB    CLRTI
0196 D2DC           344                SETB    TIRUN               ;Enable timer I.
0198 D2AB           345                SETB    ETI                 ;Turn on timer I
                                                                   ; interrupts.
019A 22             346                RET
                    347
                    348     ;This routine tries a more extreme method of bus recovery.
                    349     ; This is used if SCL or SDA are stuck and cannot
                            ; otherwise be freed.
                    350     ; (will return to the Recover routine when Timer I times out)
                    351
019B C2DE           352     BusReset:  CLR     MASTRQ              ;Release bus.
019D 7598BC         353                MOV     I2CON,#0BCh         ;Clear all I2C flags.
01A0 D2DC           354                SETB    TIRUN
01A2 80FE           355                SJMP    $                   ;Wait for timer I
                                                                   ; timeout (this will re-
                    356                                            ; set the I2C hardware).
                    357
                    358
                    359     ; This routine attempts to regain control of the I2C
                            ;                 bus after a bus fault.

                    360     ;  Returns carry clear if successful, carry set if failed.
                    361
01A4 C2DE           362     FixBus:    CLR     MastRQ              ;Turn off I2C functions.
01A6 D3             363                SETB    C
01A7 D280           364                SETB    SCL                 ;Insure I/O port is not
                                                                   ; locking I2C.
01A9 D281           365                SETB    SDA
01AB 308029         366                JNB     SCL,FixBusEx        ;If SCL is low, bus
                                                                   ; cannot be 'fixed'.
01AE 208113         367                JB      SDA,RStop           ;If SCL & SDA are high,
                                                                   ; force a stop.
01B1 752109         368                MOV     BitCnt,#9           ;Set max # of tries to
                                                                   ; clear bus.
01B4 C280           369     ChekLoop:  CLR     SCL                 ;Force an I2C clock.
01B6 31D8           370                ACALL   SDelay
01B8 208109         371                JB      SDA,RStop           ;Did it work?
01BB D280           372                SETB    SCL
01BD 31D8           373                ACALL   SDelay
01BF D521F2         374                DJNZ    BitCnt,ChekLoop     ;Repeat clocks until
                                                                   ; either SDA clears or
```

```
                    375                                     ; we run out of tries.
01C2 8013           376             SJMP    FixBusEx        ;Failed to fix bus by
                                                            ; this method.
                    377
01C4 C281           378     RStop:  CLR     SDA             ;Try forcing a stop
                                                            ; since SCL & SDA
01C6 31D8           379             ACALL   SDelay          ; are both high.
01C8 D280           380             SETB    SCL
01CA 31D8           381             ACALL   SDelay
01CC D281           382             SETB    SDA
01CE 31D8           383             ACALL   SDelay
01D0 308004         384             JNB     SCL,FixBusEx    ;Are SCL & SDA still
                                                            ; high? If so, assume bus
01D3 308101         385             JNB     SDA,FixBusEx    ; is now OK, and return
01D6 C3             386             CLR     C               ; with carry cleared.
01D7 22             387     FixBusEx: RET
                    388
                    389
                    390     ; Short delay routine (10 machine cycles).
                    391
01D8 00             392     SDelay: NOP
01D9 00             393             NOP
01DA 00             394             NOP
01DB 00             395             NOP
01DC 00             396             NOP
01DD 00             397             NOP
01DE 00             398             NOP
01DF 00             399             NOP
01E0 22             400             RET
                    401
                    402     ;*************************************************************
                    404     ;                       Main Program
                    405     ;*************************************************************
                    406
01E1 758107         407     Reset:  MOV     SP,#07h         ;Set stack location.
01E4 D2AB           408             SETB    ETI             ;Enable timer I interrupts.
01E6 D2AF           409             SETB    EA              ;Enable global interrupts.
01E8 75290B         410             MOV     XmtDat,#11      ;Set up transmit data.
01EB 752A16         411             MOV     XmtDat+1,#22    ;Set up transmit data.
01EE 752B2C         412             MOV     XmtDat+2,#44    ;Set up transmit data.
01F1 752C58         413             MOV     XmtDat+3,#88    ;Set up transmit data.
01F4 752500         414             MOV     RcvDat,#0       ;Clear receive data.
01F7 752600         415             MOV     RcvDat+1,#0     ;Clear receive data.
01FA 752700         416             MOV     RcvDat+2,#0     ;Clear receive data.
01FD 752800         417             MOV     RcvDat+3,#0     ;Clear receive data.
                    418
0200 752348         419     MainLoop: MOV   SlvAdr,#48h     ;Set slave address
                                                            ; (8-bit I/O port).
0203 752201         420             MOV     ByteCnt,#1      ;Set up byte count.
0206 1127           421             ACALL   SendData        ;Send data to slave.
0208 2002F5         422             JB      Retry,MainLoop
                    423
020B 752201         424     ML2:    MOV     ByteCnt,#1      ;Set up byte count.
020E 114E           425             ACALL   RcvData         ;Read data from slave.
0210 2002F8         426             JB      Retry,ML2
                    427
0213 7523A0         428     SL1:    MOV     SlvAdr,#0A0h    ;Set slave address
                                                            ; (RAM chip).
0216 752400         429             MOV     SubAdr,#0h      ;Set slave subaddress.
0219 752204         430             MOV     ByteCnt,#4      ;Set up byte count.
021C 1188           431             ACALL   SendSub
021E 2002F2         432             JB      Retry,SL1
                    433
```

Using the 8XC751 microcontroller as an I²C bus master

AN422

```
0221 752204          434    SL2:        MOV     ByteCnt,#4      ;Set up byte count.
0224 11B9            435                ACALL   RcvSub
0226 2002F8          436                JB      Retry,SL2
                     437
0229 0529            438                INC     XmtDat
022B 052A            439                INC     XmtDat+1
022D 052B            440                INC     XmtDat+2
022F 052C            441                INC     XmtDat+3
0231 80CD            442                SJMP    MainLoop        ;Do it all again.
                     443
                     444                ENDASSEMBLY COMPLETE, 0 ERRORS FOUND

I2CAPP          83C751 Single Master I2C Routines
ACC. . . . . . . . . . . . . . . D ADDR  00E0H  PREDEFINED
ATN. . . . . . . . . . . . . . . B ADDR  009EH  PREDEFINED
BCARL. . . . . . . . . . . . . .   NUMB  0010H
BCDR . . . . . . . . . . . . . .   NUMB  0020H
BCSTP. . . . . . . . . . . . . .   NUMB  0004H
BCSTR. . . . . . . . . . . . . .   NUMB  0008H
BCXA . . . . . . . . . . . . . .   NUMB  0080H
BIDLE. . . . . . . . . . . . . .   NUMB  0040H  NOT USED
BITCNT . . . . . . . . . . . . . D ADDR  0021H
BMRQ . . . . . . . . . . . . . .   NUMB  0040H
BTIR . . . . . . . . . . . . . .   NUMB  0010H
BUSRESET . . . . . . . . . . . . C ADDR  019BH
BXSTP. . . . . . . . . . . . . .   NUMB  0001H
BXSTR. . . . . . . . . . . . . .   NUMB  0002H
BYTECNT. . . . . . . . . . . . . D ADDR  0022H
CHEKLOOP . . . . . . . . . . . . C ADDR  01B4H
CLRINT . . . . . . . . . . . . . C ADDR  0026H
CLRTI. . . . . . . . . . . . . . B ADDR  00DDH  PREDEFINED
CTVAL. . . . . . . . . . . . . .   NUMB  0002H
DRDY . . . . . . . . . . . . . . B ADDR  009DH  PREDEFINED
EA . . . . . . . . . . . . . . . B ADDR  00AFH  PREDEFINED
ETI. . . . . . . . . . . . . . . B ADDR  00ABH  PREDEFINED
FAULT. . . . . . . . . . . . . . B ADDR  0001H
FIXBUS . . . . . . . . . . . . . C ADDR  01A4H
FIXBUSEX . . . . . . . . . . . . C ADDR  01D7H
FLAGS. . . . . . . . . . . . . . D ADDR  0020H
I2CFG. . . . . . . . . . . . . . D ADDR  00D8H  PREDEFINED
I2CON. . . . . . . . . . . . . . D ADDR  0098H  PREDEFINED
I2DAT. . . . . . . . . . . . . . D ADDR  0099H  PREDEFINED
MAINLOOP . . . . . . . . . . . . C ADDR  0200H
MASTER . . . . . . . . . . . . . B ADDR  0099H  PREDEFINED
MASTRQ . . . . . . . . . . . . . B ADDR  00DEH  PREDEFINED
ML2. . . . . . . . . . . . . . . C ADDR  020BH
NOACK. . . . . . . . . . . . . . B ADDR  0000H
P0 . . . . . . . . . . . . . . . D ADDR  0080H  PREDEFINED
RBIT . . . . . . . . . . . . . . C ADDR  0153H
RCVBYTE. . . . . . . . . . . . . C ADDR  014FH
RCVDAT . . . . . . . . . . . . . D ADDR  0025H
RCVDATA. . . . . . . . . . . . . C ADDR  004EH
RCVSUB . . . . . . . . . . . . . C ADDR  00B9H
RDACK. . . . . . . . . . . . . . C ADDR  0143H
```

Using the 8XC751 microcontroller as an I²C bus master

AN422

```
RDAT . . . . . B ADDR  009FH  PREDEFINED
RDATERR. . . . . C ADDR  0086H
RDERR. . . . . C ADDR  0163H
RDEX . . . . . C ADDR  0083H
RDLAST . . . . . C ADDR  0074H
RDLOOP . . . . . C ADDR  006AH
RECOVER. . . . . C ADDR  018AH
REPSTART . . . . . C ADDR  017AH
RESET. . . . . C ADDR  01E1H
RETRY. . . . . B ADDR  0002H
RSEX . . . . . C ADDR  0107H
RSLAST . . . . . C ADDR  00F8H
RSLOOP . . . . . C ADDR  00EEH
RSTOP. . . . . C ADDR  01C4H
RSUBERR. . . . . C ADDR  010AH
SAERR. . . . . C ADDR  011DH
SCL. . . . . B ADDR  0080H
SDA. . . . . B ADDR  0081H
SDATERR. . . . . C ADDR  004CH
SDELAY . . . . . C ADDR  01D8H
SDEX . . . . . C ADDR  0049H
SDLOOP . . . . . C ADDR  003CH
SENDAD2. . . . . C ADDR  0115H
SENDADDR . . . . . C ADDR  010CH
SENDDATA . . . . . C ADDR  0027H
SENDSTOP . . . . . C ADDR  0166H
SENDSUB. . . . . C ADDR  0088H
SL1. . . . . C ADDR  0213H
SL2. . . . . C ADDR  0221H
SLVADR . . . . . D ADDR  0023H
SP . . . . . D ADDR  0081H  PREDEFINED
SSEX . . . . . C ADDR  00B4H
SSLOOP . . . . . C ADDR  00A7H
SSUBERR. . . . . C ADDR  00B7H
STACKSAVE. . . . . D ADDR  002DH
SUBADR . . . . . D ADDR  0024H
TIMERI . . . . . C ADDR  001BH  NOT USED
TIRJN. . . . . B ADDR  00DCH  PREDEFINED
XMBIT. . . . . C ADDR  0128H
XMBIT2 . . . . . C ADDR  012AH
XMBX . . . . . C ADDR  013FH
XMERR. . . . . C ADDR  0140H
XMITADDR . . . . . C ADDR  0120H
XMITBYTE . . . . . C ADDR  0125H
XMTDAT . . . . . D ADDR  0029H
```

Software driven serial communication routines for the 83C751 and 83C752 microcontrollers

AN423

DESCRIPTION

The need often arises to make use of a serial port in connection with a microcontroller that does not have a hardware UART on-chip. Aside from the obvious cases where the microcontroller application intrinsically requires RS-232 communications to achieve its purpose, a serial output may often be a simple and convenient method of providing detailed diagnostic information to the outside world while using only a single I/O port pin. In many cases, the solution may be to implement the UART function in software. The routines included here demonstrate a method to add such a function to a microcontroller without the benefit of a hardware UART.

Examples of microcontrollers that do not have on-chip UARTs are the 83C751 and 83C752. While it is possible to connect an external UART chip to these microcontrollers, it tends to use up many I/O port pins and begins to become less economical than simply using a standard 80C51. The are several factors to be considered in deciding if the software UART method will be usable in a particular application. The first is whether the serial communication channel is to be simplex (transmit only or receive only), half-duplex (transmit and receive, but not simultaneously), or full-duplex (simultaneous transmit and receive). Both simplex and half-duplex operation are fairly easy to implement in software on an 80C51-type microcontroller, and will be covered by this application note. Full-duplex operation is more difficult to implement in software and can use up a large portion of the microcontroller's time and resources.

A second consideration to be taken into account is the amount of system resources that will be "used up" by the serial communication software. First of all, such software routines will almost always require the use of at least one counter/timer to generate the time slices for the serial bit cells. Next, the physical connection to the outside world will require one I/O port pin each for the serial input and the serial output. Moreover, the port pin used for serial input should be an external interrupt input pin. This allows the software to be interrupted automatically at the beginning of an incoming start bit and synchronizes the timer accurately to the serial data stream. Additional port pins may be used to implement signals such as Request to Send (RTS), Clear to Send (CTS), etc.

Finally, serial communication software will take up a certain amount of CPU time, more than would be required to operate a hardware UART. The overhead of software implemented serial communication may or may not be an issue, depending on the application, the throughput of the serial channel(s), the baud rate, other tasks the CPU is handling and how time-critical they are, etc.

The program listing that is included here is a demonstration of half-duplex serial routines on the 83C751 or 83C752 microcontrollers. The operation of the software would be the same on other 80C51 derivatives, except that the counter/timer operation is slightly different. The program, as listed, will send a canned message to the serial output (port pin P1.0 in this case), then wait for data on the serial input (port pin P1.5/INT0). When a character has been received on the serial input, it will be echoed through the serial output. Since the software is inherently half-duplex, the rate at which characters are received must be less than half the rate that would be possible on a full-duplex channel. This example has been set up to receive and transmit at 9600 baud when run with a 16 MHz crystal.

The operation of the routines is fairly straightforward. Beginning with a start bit occurring on the serial input line, an interrupt (external interrupt 0) will occur. At the interrupt service routine Int0, the counter/timer is loaded with a value that will result in a time delay that is approximately equivalent to half a bit cell time for the baud rate being used, less some constant to account for the elapsed time between a timer interrupt and the point where the serial input is actually sampled. The timer reload register is loaded with a value that will result in a time delay that is as close as can be calculated to one full bit cell time. The program then starts the timer and simply returns to the main program, waiting for the timer to time out, generating another interrupt.

At that point, the serial start bit should be about halfway through its nominal duration. When the first timer interrupt occurs, the timer interrupt routine Timr0 calls the receive bit routine RxBit which checks to make sure that the start bit is still valid and flags an error if it is not. The RxBit routine will then return control to the main program routine, waiting for the next timer interrupt.

On the second timer interrupt, the RxBit routine reads the serial input line and shifts the value into the serial holding register RxDat. This process is repeated until 8 bits have been read in on consecutive timer interrupts. Finally, on the tenth timer interrupt, the receive routine looks for a valid stop bit and flags an error if one is not detected. At this point, the RcvRdy flag is set to inform the main program that a character is waiting in the holding register.

The transmit routine works in a somewhat similar fashion, beginning with a call to the byte transmit routine XmtByte, which first checks to make sure that a byte receive operation is not already in progress. The RSXmt routine will then set up the timer and timer reload registers to correspond to one bit cell time, start the timer, and assert a start bit.

At each subsequent timer interrupt, the routine TxBit shifts out the next bit from the transmit holding register XmtDat, until all 8 bits have been transmitted. Once all of the data has been sent, the stop bit is asserted on the next timer interrupt. A final timer interrupt is required to insure that the stop bit lasts at least one full bit cell time. At this point, transmit flag TxFlag is cleared in order to inform the main program that the transmission is completed.

A few other useful routines are embedded in the sample program: PrByte, which converts a byte of data to hexadecimal form and transmits it; HexAsc, which converts one nibble of raw data to hexadecimal form; and Mess, which transmits an absolute string of data (usually a text message) which is terminated by a 0 byte.

This demonstration of software driven serial port routines uses 5 bytes of microcontroller RAM, two port bits (including one external interrupt input), one counter/timer, and about 256 bytes of code space, excluding the message string at the end of the listing.

```
RS751          Half-Duplex Serial Communication Routines                                    11/14/89

                ;*****************************************************************************

                ;        Software Driven Half-Duplex Serial Communication Routines
                ;              for 83C751 and 83C752 series Microcontrollers

                ;

                ;*****************************************************************************

                $Title(Half-Duplex Serial Communication Routines)
                $Date(11/14/89)
                $MOD751

                ;*****************************************************************************

FF75            BaudVal  EQU      -139                ;Timer value for 9600 baud @ 16 MHz.
                                                      ;(one bit cell time)
FFD9            StrtVal  EQU      -39                 ;Timer value to start receive.
                                                      ;(half of one bit cell time, minus the
                                                      ;time it takes the code to sample RxD)

0010            XmtDat   DATA     10h                 ;Data for RS-232 transmit routine.
0011            RcvDat   DATA     11h                 ;Data from RS-232 receive routine.
0012            BitCnt   DATA     12h                 ;RS-232 transmit & receive bit count.
0013            LoopCnt  DATA     13h                 ;Loop counter for test routine.

0020            Flags    DATA     20h
0000            TxFlag   BIT      Flags.0             ;Receive-in-progress flag.
0001            RxFlag   BIT      Flags.1             ;Transmit-in-progress flag.
0002            RxErr    BIT      Flags.2             ;Receiver framing error.
0003            RcvRdy   BIT      Flags.3             ;Receiver ready flag.

0090            TxD      BIT      P1.0                ;Port bit for RS-232 transmit.
0095            RxD      BIT      P1.5                ;Port bit for RS-232 receive (INT0).

                ;*****************************************************************************

                ; Interrupt Vectors

0000                     ORG      0                   ;Reset vector.
0000 0124                AJMP     Reset

0003                     ORG      03H                 ;External interrupt 0.
0003 019F                AJMP     ExInt0              ;Indicates RS-232 start bit received.

000B                     ORG      0BH                 ;Timer 0 interrupt.
000B 0175                AJMP     Timr0               ;Baud rate generator.

0013                     ORG      13H                 ;External interrupt 1 (not used).
0013 32                  RETI

001B                     ORG      1BH                 ;Timer I interrupt (not used).
001B 32                  RETI

0023                     ORG      23H                 ;I2C interrupt (not used).
0023 32                  RETI
```

```
                ;*************************************************************************

                ;Simple test of RS-232 transmit and receive.

0024 758130     Reset:  MOV     SP,#30h
0027 752000             MOV     Flags,#0          ;Clear RS-232 flags.
002A C201               CLR     RxFlag
002C 758800             MOV     TCON,#00h         ;Set up timer controls.
002F 75A882             MOV     IE,#82h           ;Enable timer 0 interrupts.

0032 751310             MOV     LoopCnt,#16       ;Test transmit first.
0035 7900               MOV     R1,#0             ;Zero line count.
0037 90010C             MOV     DPTR,#Msg1        ;Point to message string.
003A 11FB       Loop1:  ACALL   Mess              ;Send an RS-232 message repeatedly.
003C 743A               MOV     A,#':'
003E 1154               ACALL   XmtByte
0040 E9                 MOV     A,R1
0041 11DD               ACALL   PrByte            ;Print R1 contents.
0043 09                 INC     R1                ;Advance R1 value.
0044 D513F3             DJNZ    LoopCnt,Loop1

0047 D2A8       Loop2:  SETB    EX0               ;Enable interrupt 0 (RS-232 receive).
0049 3003FD             JNB     RcvRdy,$          ;Wait for data available.
004C C203               CLR     RcvRdy
004E E511               MOV     A,RcvDat          ;Echo same byte.
0050 1154               ACALL   XmtByte
0052 80F3               SJMP    Loop2


                ; Send a byte out RS-232 and wait for completion before returning.
                ; (use if there is nothing else to do while RS-232 is busy)

0054 2001FD     XmtByte: JB     RxFlag,$          ;Wait for receive complete.
0057 115D               ACALL   RSXmt             ;Send ACC to RS-232 output.
0059 2000FD             JB      TxFlag,$          ;Wait for transmit complete.
005C 22                 RET


                ; Begin RS-232 transmit.

005D F510       RSXmt:  MOV     XmtDat,A          ;Save data to be transmitted.
005F 75120A             MOV     BitCnt,#10        ;Set bit count.
0062 758CFF             MOV     TH,#High BaudVal  ;Set timer for baud rate.
0065 758A75             MOV     TL,#Low BaudVal
0068 758DFF             MOV     RTH,#High BaudVal ;Also set timer reload value.
006B 758B75             MOV     RTL,#Low BaudVal
006E D28C               SETB    TR                ;Start timer.
0070 C290               CLR     TxD               ;Begin start bit.
0072 D200               SETB    TxFlag            ;Set transmit-in-progress flag.
0074 22                 RET


                ; Timer 0 timeout: RS-232 receive bit or transmit bit.

0075 C0E0       Timr0:  PUSH    ACC
0077 C0D0               PUSH    PSW
0079 20013E             JB      RxFlag,RxBit      ;Is this a receive timer interrupt?
007C 200007             JB      TxFlag,TxBit      ;Is this a transmit timer interrupt?
```

```
007F C28C          T0Ex1:   CLR     TR                  ;Stop timer.
0081 D0D0          T0Ex2:   POP     PSW
0083 D0E0                   POP     ACC
0085 32                     RETI


                   ; RS-232 transmit bit routine.

0086 D51204        TxBit:   DJNZ    BitCnt,TxBusy       ;Decrement bit count, test for done.
0089 C200                   CLR     TxFlag              ;End of stop bit, release timer.
008B 80F2                   SJMP    T0Ex1               ;Stop timer and exit.

008D E512          TxBusy:  MOV     A,BitCnt            ;Get bit count.
008F B40104                 CJNE    A,#1,TxNext         ;Is this a stop bit?
0092 D290                   SETB    TxD                 ;Set stop bit.
0094 80EB                   SJMP    T0Ex2               ;Exit.

0096 E510          TxNext:  MOV     A,XmtDat            ;Get data.
0098 13                     RRC     A                   ;Advance to next bit.
0099 F510                   MOV     XmtDat,A
009B 9290                   MOV     TxD,C               ;Send data bit.
009D 80E2                   SJMP    T0Ex2               ;Exit.


                   ;Begin RS-232 receive (after external interrupt 0).

009F 75120A        ExInt0:  MOV     BitCnt,#10          ;Set receive bit count.
00A2 758CFF                 MOV     TH,#High StrtVal    ;First timeout in HALF a bit time.
00A5 758AD9                 MOV     TL,#Low StrtVal
00A8 758DFF                 MOV     RTH,#High BaudVal   ;Set timer reload for baud rate.
00AB 758B75                 MOV     RTL,#Low BaudVal
00AE 751100                 MOV     RcvDat,#0           ;Initialize received data to 0.
00B1 C2A8                   CLR     EX0                 ;Disable external interrupt 0.
00B3 C202                   CLR     RxErr               ;Clear error flag.
00B5 D28C                   SETB    TR                  ;Start timer.
00B7 D201                   SETB    RxFlag              ;Set receive-in-progress flag.
00B9 32                     RETI


                   ; RS-232 receive bit routine.

00BA D5120D        RxBit:   DJNZ    BitCnt,RxBusy       ;Decrement bit count, test for stop.
00BD 209502                 JB      RxD,RxBitEx         ;Valid stop bit?
00C0 D202          RxBtErr: SETB    RxErr               ;Bad stop bit, tell mainline.
00C2 C201          RxBitEx: CLR     RxFlag              ;Release timer for other purposes.
00C4 D2A8                   SETB    EX0                 ;Re-enable external interrupt 0.
00C6 D203                   SETB    RcvRdy              ;Tell mainline that a byte is ready.
00C8 80B5                   SJMP    T0Ex1               ;Stop timer and exit.

00CA E512          RxBusy:  MOV     A,BitCnt            ;Get bit count.
00CC B40905                 CJNE    A,#9,RxNext         ;Is this a start bit?
00CF 2095EE                 JB      RxD,RxBtErr         ;Valid start bit?
00D2 80AD                   SJMP    T0Ex2               ;Exit.

00D4 E511          RxNext:  MOV     A,RcvDat            ;Get partial receive byte.
00D6 A295                   MOV     C,RxD               ;Get receive pin value.
00D8 13                     RRC     A                   ;Shift in new bit.
00D9 F511                   MOV     RcvDat,A            ;Save updated receive byte.
```

```
00DB 80A4                 SJMP     T0Ex2              ;Exit.


                  ; Print byte routine: print ACC contents as ASCII hexadecimal.

00DD C0E0         PrByte: PUSH   ACC
00DF C4                   SWAP   A
00E0 11EB                 ACALL  HexAsc
00E2 1154                 ACALL  XmtByte
00E4 D0E0                 POP    ACC
00E6 11EB                 ACALL  HexAsc            ;Print nibble in ACC as ASCII hex.
00E8 1154                 ACALL  XmtByte
00EA 22                   RET


                  ; Hexadecimal to ASCII conversion routine.

00EB 540F         HexAsc: ANL    A,#0FH            ;Convert a nibble to ASCII hex.
00ED 30E308               JNB    ACC.3,NoAdj
00F0 20E203               JB     ACC.2,Adj
00F3 30E102               JNB    ACC.1,NoAdj
00F6 2407         Adj:    ADD    A,#07H
00F8 2430         NoAdj:  ADD    A,#30H
00FA 22                   RET


                  ; Message string transmit routine.

00FB C0E0         Mess:   PUSH     ACC
00FD 7800                 MOV      R0,#0              ;R0 is character pointer (string
00FF E8           Mesl:   MOV      A,R0               ; length is limited to 256 bytes).
0100 93                   MOVC     A,@A+DPTR          ;Get byte to send.
0101 B40003               CJNE     A,#0,Send          ;End of string is indicated by a 0.
0104 D0E0                 POP      ACC
0106 22                   RET

0107 1154         Send:   ACALL    XmtByte            ;Send a character.
0109 08                   INC      R0                 ;Next character.
010A 80F3                 SJMP     Mesl

010C 0D0A         Msgl:   DB       0Dh, 0Ah
010E 54686973             DB       'This is a test of the software serial routines.', 0
0112 20697320
0116 61207465
011A 7374206F
011E 66207468
0122 6520736F
0126 66747761
012A 72652073
012E 65726961
0132 6C20726F
0136 7574696E
013A 65732E00

                          END

ASSEMBLY COMPLETE, 0 ERRORS FOUND
```

Software driven serial communication routines for the 83C751 and 83C752 microcontrollers

AN423

```
ACC. . . . . D ADDR  00E0H  PREDEFINED
ADJ. . . . . C ADDR  00F6H
BAUDVAL. . . . . NUMB  FF75H
BITCNT . . . . . D ADDR  0012H
EX0. . . . . B ADDR  00A8H  PREDEFINED
EXINT0 . . . . . C ADDR  009FH
FLAGS. . . . . D ADDR  0020H
HEXASC . . . . . C ADDR  00EBH
IE . . . . . D ADDR  00A8H  PREDEFINED
LOOP1. . . . . C ADDR  003AH
LOOP2. . . . . C ADDR  0047H
LOOPCNT. . . . . D ADDR  0013H
MESL . . . . . C ADDR  00FFH
MESS . . . . . C ADDR  00FBH
MSG1 . . . . . C ADDR  010CH
NOADJ. . . . . C ADDR  00F8H
P1 . . . . . D ADDR  0090H  PREDEFINED
PRBYTE . . . . . C ADDR  00DDH
PSW. . . . . D ADDR  00D0H  PREDEFINED
RCVDAT . . . . . D ADDR  0011H
RCVRDY . . . . . B ADDR  0003H
RESET. . . . . C ADDR  0024H
RSXMT. . . . . C ADDR  005DH
RTH. . . . . D ADDR  008DH  PREDEFINED
RTL. . . . . D ADDR  008BH  PREDEFINED
RXBIT. . . . . C ADDR  00BAH
RXBITEX. . . . . C ADDR  00C2H
RXBTERR. . . . . C ADDR  00C0H
RXBUSY . . . . . C ADDR  00CAH
RXD. . . . . B ADDR  0095H
RXERR. . . . . B ADDR  0002H
RXFLAG . . . . . B ADDR  0001H
RXNEXT . . . . . C ADDR  00D4H
SEND . . . . . C ADDR  0107H
SP . . . . . D ADDR  0081H  PREDEFINED
STRTVAL. . . . . NUMB  FFD9H
T0EX1. . . . . C ADDR  007FH
T0EX2. . . . . C ADDR  0081H
TCON . . . . . D ADDR  0088H  PREDEFINED
TH . . . . . D ADDR  008CH  PREDEFINED
TIMR0. . . . . C ADDR  0075H
TL . . . . . D ADDR  008AH  PREDEFINED
TR . . . . . B ADDR  008CH  PREDEFINED
TXBIT. . . . . C ADDR  0086H
TXBUSY . . . . . C ADDR  008DH
TXD. . . . . B ADDR  0090H
TXFLAG . . . . . B ADDR  0000H
TXNEXT . . . . . C ADDR  0096H
XMTBYTE. . . . . C ADDR  0054H
XMTDAT . . . . . D ADDR  0010H
```

8051 family warm boot determinations AN424

DESCRIPTION

For some classes of applications, it may be desirable to know if the application of the reset signal to a microcontroller is due to an initial power-on sequence, or is the result of an external signal such as an operator pressing a reset pushbutton, or the result of a watchdog timer or similar event.

While there are perhaps numerous hardware solutions that can be employed, a simple software solution can offer a high degree of confidence in making this determination. The task is to determine the differences in state of resources internal to the microcontroller that would occur as a result of these two types of reset conditions. With respect to the 80C51 family of microcontrollers, on-chip resources consist of the special function registers (SFRs) and the internal data memory (RAM). Most of the SFR locations are initialized as a result of a reset condition and thus cannot be used for this determination. The data memory contents are unaffected by reset. Thus, valid data loaded into the RAM of the 80C51 while executing a program would not be affected by the application of an external reset signal provided the power source for the microcontroler has not been removed (as is the case for a "warm boot").

The contents of data memory as a result of an initial application of power, however, is indeterminate. While this effect has not been extensively characterized, empirical observation suggests that it is highly random in nature. If it is assumed, for the moment, that the behavior of a given byte of data memory is such that it will power-up with a value that is totally random, then there is a one in eight chance that it will power-up with a predetermined value. If the assumption is extended to two bytes, a 16-bit number, then there is one in 2^{16} chance that both bytes will power-up with predetermined values. Extending this to four bytes results in a one in 2^{32} chance; a very small probability. This is the basis for the software determination of a warm or cold boot condition.

The technique consists of evaluating the contents of four consecutive bytes of data memory following a reset condition to determine whether these bytes had been previously loaded with known data values. If the contents of all four bytes match predetermined values, this is interpreted to be a warm boot condition. If there is no match, it is then interpreted to be a cold boot condition. At this point, it is necessary to load these four bytes with predetermined data to prepare for the possibility of a subsequent warm boot condition.

The software example included in this application brief can be used to perform this warm or cold boot determination.

The symbols WARM1 thorough WARM4 represent the predetermined values. The symbol WARM is the address of the first of the four consecutive bytes in data memory. It is set to 30H to avoid conflict with the four register banks, the stack, and the bit-addressable locations in data memory. The symbol WARMBT is a bit-addressable location used as a status bit. It is set as the result of a warm boot and cleared as a result of a cold boot.

The label START is the location of the first instruction to be executed following a reset (address = 0000H). An instruction is located here to jump into the main body of the program to bypass the interrupt vector locations.

The main program body begins by loading register R0 with the address of the first byte in data memory to be evaluated. The contents of this first byte is compared with the first predetermined value. If there is no match, the conclusion is that it is a cold boot. However, if a match is found, this does not imply that it is a warm boot since all four bytes must match, and therefore the remaining three bytes must also be evaluated. Register R0 is incremented to point to the second byte and then compared to the second predetermined value. Comparison of the bytes proceeds until either a no match condition is found or until all four bytes have been evaluated successfully. If all four bytes compared favorable, then a status bit (WARMBT) is set to indicate a warm boot and the remainder of the application program is completed.

An unsuccessful comparison results in branching to the label COLD. This section of code clears the status bit (WARMBT) to indicate a cold boot, and loads the four bytes of data memory with the predetermined values preparing the system for a subsequent possible warm boot. Program flow then continues with the remainder of the application program.

8051 family warm boot determinations AN424

```
                        1
                        2    ;warm boot application example
                        3
  0030                  4    WARM    EQU     30H             ;first location of the four bytes in RAM
  0055                  5    WARM1   EQU     55H             ;first predetermined value
  00AA                  6    WARM2   EQU     0AAH            ;second predetermined value
  0033                  7    WARM3   EQU     33H             ;third predetermined value
  00CC                  8    WARM4   EQU     0CCH            ;fourth predetermined value
  0000                  9    WARMBT  EQU     0               ;warm boot status bit
                       10
0000                   11            ORG     0
                       12
0000 020026            13    START:  JMP     MAIN            ;bypass interrupt vectors
                       14
0026                   15            ORG     26H
                       16
0026 7830              17    MAIN:   MOV     R0,#WARM        ;pointer for first byte
0028 B65511            18            CJNE    @R0,#WARM1,COLD ;test first byte
002B 08                19            INC     R0              ;pointer for second byte
002C B6AA0D            20            CJNE    @R0,#WARM2,COLD ;test second byte
002F 08                21            INC     R0              ;pointer for third byte
0030 B63309            22            CJNE    @R0,#WARM3,COLD ;test third byte
0033 08                23            INC     R0              ;pointer for fourth byte
0034 B6CC05            24            CJNE    @R0,#WARM4,COLD ;test fourth byte
0037 D200              25            SETB    WARMBT          ;this is a warm start
0039 02004B            26            JMP     INIT            ;continue with rest of application
003C C200              27    COLD:   CLR     WARMBT          ;this is a cold boot
003E 7830              28            MOV     R0,#WARM        ;pointer for first byte
0040 7655              29            MOV     @R0,#WARM1      ;load the four bytes for future test
0042 08                30            INC     R0              ;
0043 76AA              31            MOV     @R0,#WARM2      ;
0045 08                32            INC     R0              ;
0046 7633              33            MOV     @R0,#WARM3      ;
0048 08                34            INC     R0              ;
0049 76CC              35            MOV     @R0,#WARM4      ;
004B                   36    INIT:                           ;continue with the application
                       37
                       38    END

ASSEMBLY COMPLETE, 0 ERRORS FOUND

COLD . . . . . C ADDR  003CH
INIT . . . . . C ADDR  004BH
MAIN . . . . . C ADDR  0026H
START. . . . . C ADDR  0000H  NOT USED
WARM . . . . . NUMB  0030H
WARM1. . . . . NUMB  0055H
WARM2. . . . . NUMB  00AAH
WARM3. . . . . NUMB  0033H
WARM4. . . . . NUMB  00CCH
WARMBT . . . . . NUMB  0000H
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

DESCRIPTION

This application note shows how to use the PCD8584 I²C-bus controller with 80C51 family microcontrollers. One typical way of connecting the PCD8584 to an 80C31 is shown. Some basic software routines are described showing how to transmit and receive bytes in a single master system. An example is given of how to use these routines in an application that makes use of the I²C circuits on an I²C demonstration board.

The PCD8584 is used to interface between parallel microprocessor or microcontroller buses and the serial I²C bus. For a description of the I²C bus protocol refer to the I²C bus specification which is printed in the microcontroller user guide.

The PCD8584 controls the transmission and reception of data on the I²C bus, arbitration, clock speeds and transmission and reception of data on the parallel bus. The parallel bus is compatible with 80C51, 68000, 8085 and Z80 buses. Communication with the I²C-bus can be done on an interrupt or polled basis. This application note focuses on interfacing with 8051 microcontrollers in single master systems.

PCD8584

In Figure 1, a block diagram is shown of the PCD8584. Basically it consists of an I²C-interface similar to the one used in 84Cxx family microcontrollers, and a control block for interfacing to the microcontroller.

The control block can automatically determine whether the control signals are from 80xx or 68xxx type of microcontrollers.

This is determined after the first write action from the microcontroller to the PCD-8584. The control block also contains a programmable divider which allows the selection of different PCD8584 and I²C clocks.

The I²C interface contains several registers which can be written and read by the microcontroller.

S1 is the control/status register. This register is accessed while the A0 input is 1. The meaning of the bits depends on whether the register is written to or read from. When used as a single master system the following bits are important:

PIN: Interrupt bit. This bit is made active when a byte is sent/received to/from the I²C-bus. When ENI is made active, PIN also controls the external INT line to interrupt the microcontroller.

ES0–ES2: These bits are used as pointer for addressing S0, S0', S2 and S3. Setting ES0 also enables the Serial I/O.

ENI: Enable Interrupt bit. Setting this bit enables the generation of interrupts on the INT line.

STA, STO: These bits allow the generation of START or STOP conditions.

ACK: With this bit set and the PCD8584 is in master/receiver mode, no acknowledge is generated by the PCD8584. The slave/transmitter now knows that no more data must be sent to the I²C-bus.

BER: This bit may be read to check if bus errors have occurred.

BB: This bit may be read to check whether the bus is free for I²C-bus transmission.

S2 is the clock register. It is addressed when A0 = 0 and ES0–ES2 = 010 in the previous write cycle to S1. With the bits S24–S20 it is possible to select 5 input clock frequencies and 4 I²C clock frequencies.

S3 is the interrupt vector register. It is addressed when A0 = 0 and ES0–ES2 = 001 in the previous write cycle to S1. This register is not used when an 80C51 family microcontroller is used. An 80C51 microcontroller has fixed interrupt vector addresses.

S0' is the own address register. It is addressed when A0 = 0 and ES0–ES2 = 000. This register contains the slave address of the PCD8584. In the single master system described here, this register has no functional use. However, by writing a value to S0', the PCD8584 determines whether an 80Cxx or 68xxx type microcontroller is the controlling microcontroller by looking at the $\overline{CS}$ and $\overline{WR}$ lines. So independent of whether the PCD8584 is used as master or slave, the microcontroller should always first write a value to S0' after reset.

S0 is the I²C data register. It is addressed when A0 = 0 and ES0–ES2 = 1x0. Transmission of a byte on the I²C bus is done by writing this byte to S0. When the transmission is finished, the PIN bit in S1 is reset and if ENI is set, an interrupt will be generated. Reception of a byte is signaled by resetting PIN and by generating an interrupt if ENI is set. The received byte can be read from S0.

The SDA and SCL lines have no protection diodes to V_{DD}. This is important for multi-master systems. A system with a PCD8584 can now be switched off without causing the I²C-bus to hang-up. Other masters still can use the bus.

For more information of the PCD8584 refer to the data sheet.

PCD8584/8031 Hardware Interface

Figure 2 shows a minimum system with an 8051 family controller and a PCD8584. In this example, an 80C31 is used. However any 80C51 family controller with external addressing capability can be used.

The software resides in EPROM U3. For addressing this device, latch U2 is necessary to demultiplex the lower address bits from the data bits. The PCD8584 is mapped in the external data memory area. It is selected when A1 = 0. Because in this example no external RAM or other mapped peripherals are used, no extra address decoding components are necessary. A0 is used by the PCD8584 for proper register selection in the PCD8584.

U5A is an inverter with Schmitt trigger input and is used to buffer the oscillator signal of the microcontroller. Without buffering, the rise and fall time specifications of the CLK signal are not met. It is also important that the CLK signal has a duty cycle of 50%. If this is not possible with certain resonators or microcontrollers, then an extra flip-flop may me necessary to obtain the correct duty cycle.

U5C and U5D are used to generate the proper reset signals for the microcontroller and the PCD8584.

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

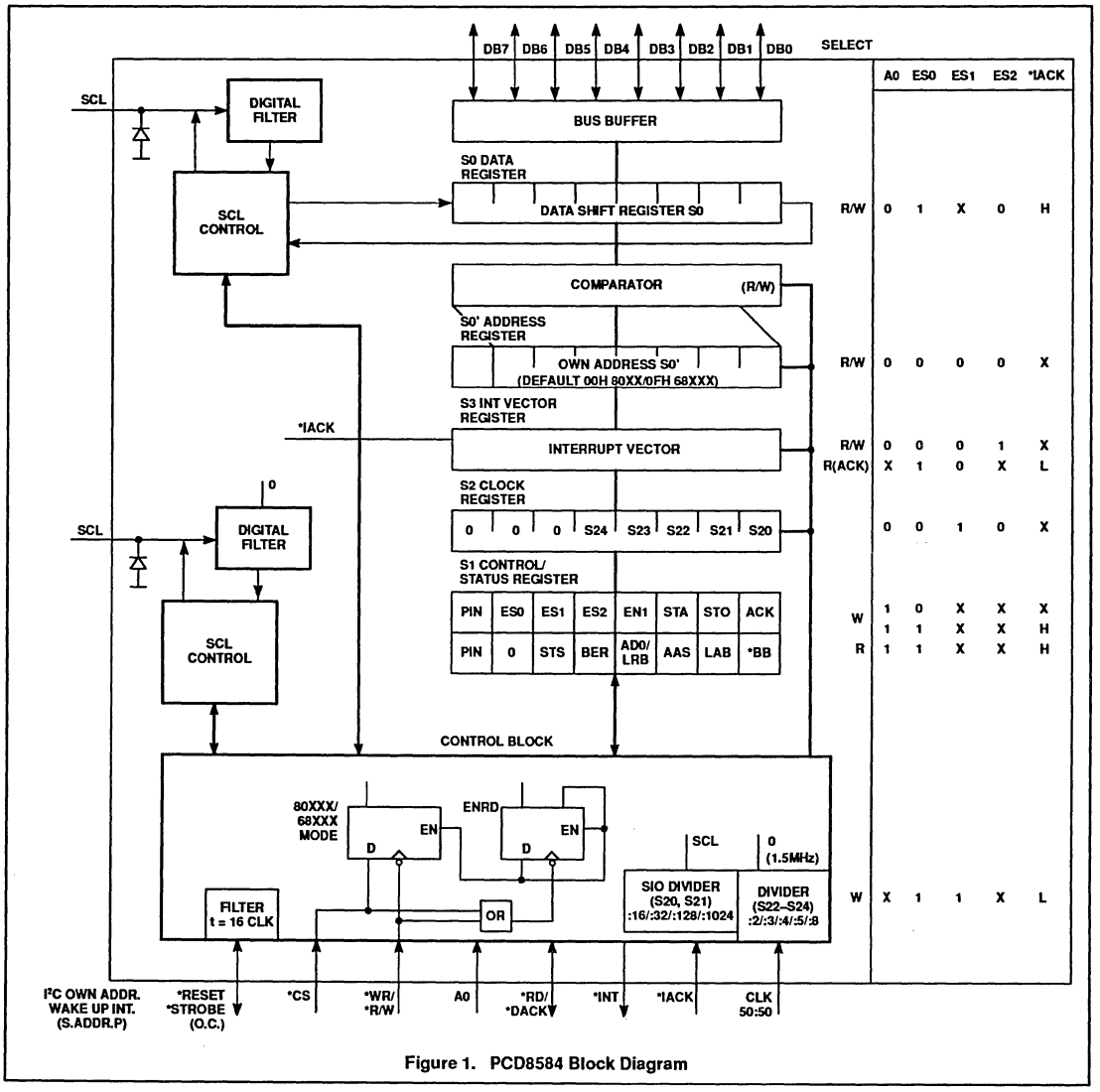


Figure 1. PCD8584 Block Diagram

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

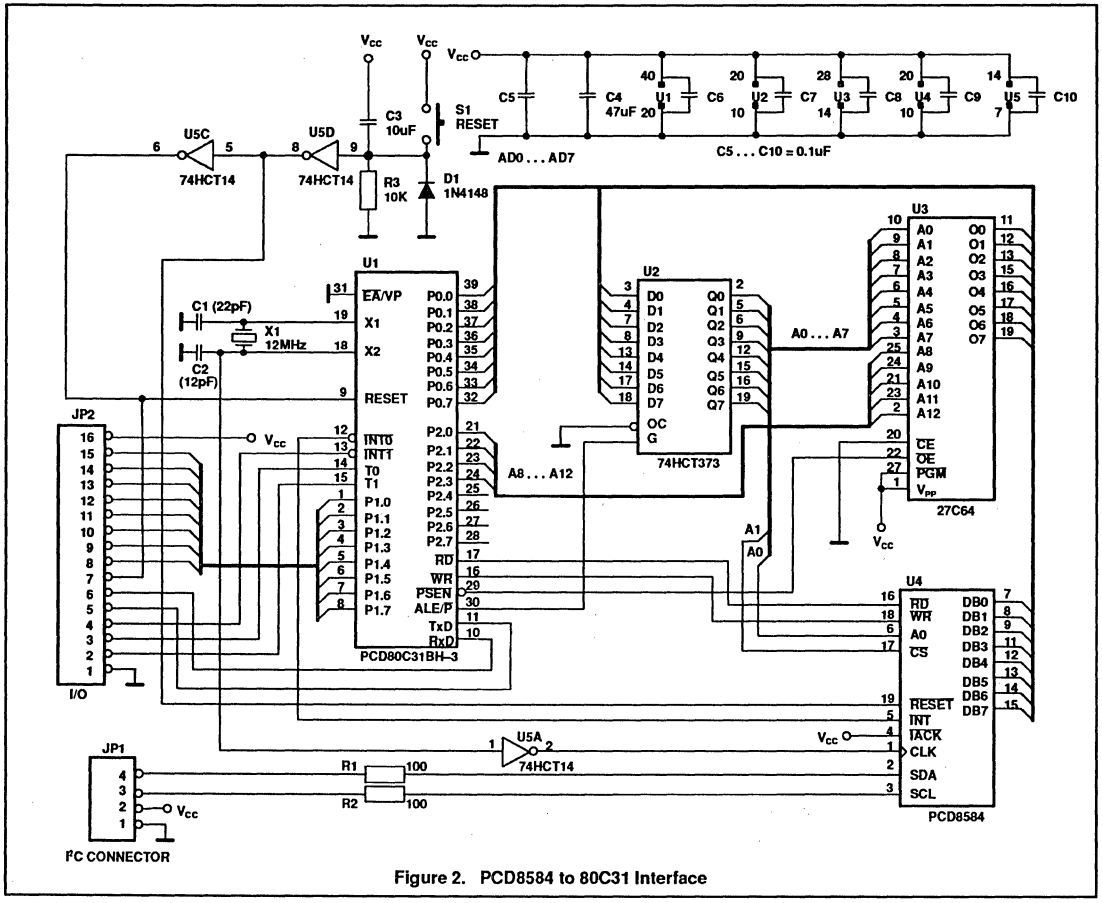


Figure 2. PCD8584 to 80C31 Interface

Basic PCD8584/8031 Driver Routines

In the listing section (page 728), some basic routines are shown. The routines are divided in two modules. The module ROUTINE contains the driver routines and initialization of the PCD8584. The module INTERR contains the interrupt handler. These modules may be linked to a module with the user program that uses the routines in INTERR and ROUTINE. In this application note, this module will be called USER. A description of ROUTINE and INTERR follows.

Module ROUTINE

Routine Sendbyte (Lines 17–20)—
This routine sends the contents of the accumulator to the PCD8584. The address is such that A0 = 0. Which register is accessed depends on the contents of ES0–ES2 of the control register. The address of the PCD8584 is in variable 'PCD8584'. This must have been previously defined in the user program. The DPTR is used as a pointer for addressing the peripheral. If the address is less than 255, then R0 or R1 may be used as the address pointer.

Routine Sendcontr (Lines 25, 26)—
This routine is similar to Sendbyte, except that now A0 = 1. This means that the contents of the accumulator are sent to the control register S1 in the PCD8584.

Routine Readbyte (Lines 30–33)—
This routine reads a register in the PCD8584 with A0 = 0. Which register depends on ES0 –ES2 of the control register. The result of the read operation is returned in the accumulator.

Routine Readcontr (Lines 37–39)—
This routine is similar to Readbyte, except that now A0 = 1. This means that the accumulator will contain the value of status register S1 of the PCD8584.

Routine Start Lines (44–56)—
This routine generates a START–condition and the slave address with a R/W bit. In line 44, the variable IIC_CNT is reset. This variable is used as a byte counter to keep track of the number of bytes that are received or transmitted. IIC_CNT is defined in module INTERR.

Lines 45–46 increment the variable NR_BYTES if the PCD8584 must receive data. NR_BYTES is a variable that indicates how many bytes have to be received or transmitted. It must be given the correct value in the USER module. Receiving or transmitting is distinguished by the value of the DIR bit. This must also be given the correct value in the USER module.

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

Then the status register of PCD8584 must be read to check if the I²C bus is free. First the status register must be addressed by giving ES0– ES2 of the control register the correct value (lines 47–48). Then the Bus Busy bit is tested until the bus is free (lines 49–50). If this is the case, the slave address is sent to data register S0 and the I2C_END bit is cleared (lines 51–53). The slave address is set by the user program in variable USER. The LSB of the slave address is the R/W bit. I2C_END can be tested by the user program whether an I2C reception/transmission is in progress or not.

Next the START condition will be generated and interrupt generation enabled by setting the appropriate bits in control register S1. (lines 54–55).

Now the routine will return back to the user program and other tasks may be performed. When the START condition, slave address and R/W bit are sent, and the ACK is received, the PCD8584 will generate an interrupt. The interrupt routine will determine if more bytes have to be received or transmitted.

Routine Stop (Lines 59–62) —
Calling this routine, a STOP condition will be sent to the I²C bus. This is done by sending the correct value to control register S1 (lines 59– 61). After this the I2C_END bit is set, to indicate to the user program that a complete I²C sequence has been received or transmitted.

Routine I2C_Init (Lines 65–76)—
This routine initializes the PCD8584. This must be done directly after reset. Lines 67–70 write data to 'own address' register S0'. First the correct address of S0' is set in control register S1 (lines 67–68), then the correct value is written to it (lines 69–70). The value for S0' is in variable SLAVE_ADR and set by the user program. As noted previously, register S0' must always be the first register to be accessed after reset, because the PCD8584 now determines whether an 80Cxxx or 68xxx microcontroller is connected. Lines 72–76 set the clock register S2. The variable I2C_CLOCK is also set by the user program.

Module INTERR
This module contains the I²C interrupt routine. This routine is called every time a byte is received or transmitted on the I²C bus. In lines 12–15 RAM space for variables is reserved.

BASE is the start address in the internal 80C51 RAM where the data is stored that is received, or where the data is stored that has to be transmitted.

NR_BYTES, IIC_CNT and SLAVE were explained earlier. I2C_END and DIR are flags that are used in the program. I2C_END indicates whether an I²C transmission or reception is in progress. DIR indicates whether the PCD8584 has to receive or transmit bytes. The interrupt routine makes use of register bank 1.

The transmission part of the routine starts at line 42. In lines 42–43, a check is made whether IIC_CNT = NR_BYTES. If true, all bytes are sent and a STOP condition may be generated (lines 44–45).

Next the pointer for the internal RAM is restored (line 46) and the byte to be transmitted is fetched from the internal RAM (line 47). Then this byte is sent to the PCD8584 and the variables are updated (lines 47–49). The interrupt routine is left and the user program may proceed. The receive part starts from line 55. First a check is made if the next byte to be received is the last byte (lines 56–59). If true the ACK must be disabled when the last byte is received. This is accomplished by resetting the ACK bit in the control register S1 (lines 60–61).

Next the received byte may be read (line 62) from data register S0. The byte will be temporary stored in R4 (line 63). Then a check is made if this interrupt was the first after a START condition. If so, the byte read has no meaning and the interrupt routine will be left (lines 68–70). However by reading the data register S0 the next read cycle is started.

If valid data is received, it will be stored in the internal RAM addressed by the value of BASE (lines 71–73). Finally a check is made if all bytes are received. If true, a STOP condition will be sent (lines 75–78).

EXAMPLES
In the listing section (starting on page 8), some examples are shown that make use of the routines described before. The examples are transmission of a sequence, reception of I²C data and an example that combines both.

The first example sends bytes to the PCD8577 LCD driver on the OM1016 demonstration board. Lines 7 to 10 define the interface with the other modules and should be included in every user program. Lines 14 to 16 define the segments in the user module. It is completely up to the user how to organize this.

Lines 24 and 28 are the reset and interrupt vectors. The actual user program starts at line 33. Here three variables are defined that are used in the I²C driver routines. Note that PCD8584 must be an even address, otherwise the wrong internal registers will be accessed! Lines 37–42 initialize the interrupt logic of the microcontroller. Next the PCD8584 will be initialized (line 45).

The PCD8584 is now ready to transmit data. A table is made in the routine at line 61. For the PCD8577, the data is a control byte and the segment data. Note that the table does not contain the slave address of the LCD driver. In lines 51–54, variables are made ready to start the transmission. This consists of defining the direction of the transmission (DIR), the address where the data table starts (BASE), the number of bytes to transmit (NR_BYTES, without slave address!) and the slave address (SLAVE) of the I²C peripheral that has to be accessed.

In line 55 the transmission is started. Once the I²C transmission is started, the user program can do other tasks because the transmission works on interrupts. In this example a loop is performed (line 58). The user can check the end of the transmission during the other tasks, by testing the I2C_END bit regularly.

The second example program receives 2 bytes from the PCF8574P I/O expander on the OM1016 demonstration board. Until line 45 the program is identical to the transmit routine because it consists of initialization and variable definition. From line 48, the variables are set for I²C reception. The received bytes are stored in RAM area from label TABLE. During reception, the user program can do other tasks. By testing the I2C_END bit the user can determine when to start processing the data in the TABLE.

The third example program displays time from the PCF8583P clock/calendar/RAM on the LCD display driven by the PCF8577. The LED display (driven by SAA1064) shows the value of the analog inputs of the A/D converter PCF8591. The four analog inputs are scanned consecutively.

In this example, both transmit and receive sequences are implemented as shown in the previous examples. The main clock part is from lines 62–128. This contains the calls to the I²C routines. From lines 135–160, routines are shown that prepare the data to be transmitted. Lines 171 to 232 are the main program for the AD converter and LED display. Lines 239 to 340 contain routines used by the main program. This demo program can also be used with the I²C peripherals on the OM1016 demonstration board.

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
ASM51  TSW  ASSEMBLER   Routines for PCD8584

LOC   OBJ          LINE  SOURCE

                      1  $TITLE (Routines for PCD8584)
                      2  $PAGELENGTH(40)
                      3  ;Program written for PCD8584 as master
                      4  ;
                      5          PUBLIC READBYTE,READCONTR,SENDBYTE
                                 PUBLIC SENDCONTR,START,STOP
                      6          PUBLIC I2C_INIT
                      7          EXTRN  BIT(I2C_END,DIR)
                      8          EXTRN  DATA(SLAVE,IIC_CNT,NR_BYTES)
                      9          EXTRN  NUMBER(SLAVE_ADR,I2C_CLOCK,PCD8584)
                     10  ;
                     11  ;Define code segment
                     12  ROUTINE    SEGMENT CODE
----                 13             RSEG    ROUTINE
                     14  ;
                     15  ;SENDBYTE sends a byte to PCD8584 with A0=0
                     16  ;Byte to be send must be in accu
0000:            R   17  SENDBYTE:
0000: 900000     R   18          MOV DPTR,#PCD8584 ;Register address
0003: F0             19  SEND:    MOVX @DPTR,A      ;Send byte
0004: 22             20          RET
                     21  ;
                     22  ;SENDCONTR sends a byte to PCD8584 with A0=1
                     23  ;Byte to be send must be in accu
0005:                24  SENDCONTR:
0005: 900001     R   25          MOV DPTR,#PCD8584+01H ;Register address
0008: 80F9           26          JMP SEND
                     27  ;
                     28  ;READBYTE reads a byte from PCD8584 with A0=0
                     29  ;Received byte is stored in accu
000A:                30  READBYTE:
000A: 900000     R   31          MOV DPTR,#PCD8584 ;Register address
000D: E0             32  REC:     MOVX A,@DPTR      ;Receive byte
000E: 22             33          RET
                     34  ;
                     35  ;READCONTR reads a byte from PCD8584 with A0=1
                     36  ;Received byte is stored in accu
000F:                37  READCONTR:
000F: 900001     R   38          MOV DPTR,#PCD8584+01H ;Register address
0012: 80F9           39          JMP REC
                     40  ;
                     41  ;START tests if the I2C bus is ready. If ready a
                     42  ;START-condition will be sent, interrupt generation
                     43  ;and acknowledge will be enabled.
0014: 750000     R   44  START:   MOV IIC_CNT,#00 ;Clear  I2C byte counter
0017: 200002     R   45          JB DIR,PROCEED  ;If DIR is 'receive' then
001A: 0500       R   46          INC NR_BYTES    ;increment NR_BYTES
001C: 7440           47  PROCEED:MOV A,#40H       ; Read STATUS register of
                                                  ; 8584
001E: 120005     R   48          CALL SENDCONTR
0021: 12000F     R   49  TESTBB:  CALL READCONTR
0024: 30E0FA         50          JNB ACC.0,TESTBB; Test BB/ bit
0027: E500       R   51          MOV A,SLAVE
0029: C200       R   52          CLR I2C_END     ;Reset I2C ready bit
002B: 120000     R   53          CALL SENDBYTE   ;Send slave address
002E: 744D           54          MOV A,#01001101B;Generate START, set ENI,
                                                 ;set ACK
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
0030: 120005   R     55          CALL SENDCONTR
0033: 22             56          RET
                     57  ;
                     58  ;STOP will generate a STOP condition and set the
                         ;I2C_END bit
0034: 74C3           59  STOP:   MOV A,#11000011B
0036: 120005   R     60          CALL SENDCONTR  ;Send STOP condition
0039: D200     R     61          SETB I2C_END    ;Set I2C_END bit
003B: 22             62          RET
                     63  ;
                     64  ;I2C_init does the initialisation of the PCD8584
003C:                65  I2C_INIT:
                     66  ;Write own slave address
003C: E4             67          CLR A
003D: 120005   R     68          CALL SENDCONTR  ;Write to control register
0040: 7400     R     69          MOV A,#SLAVE_ADR
0042: 120000   R     70          CALL SENDBYTE    ;Write to own slave
                                                  ;register
                     71  ;Write clock register
0045: 7420           72          MOV A,#20H
0047: 120005   R     73          CALL SENDCONTR  ;Write to control register
004A: 7400     R     74          MOV A,#I2C_CLOCK
004C: 120000   R     75          CALL SENDBYTE   ;Write to clock register
004F: 22             76          RET
                     77  ;
0050:                78          END
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
ASM51  TSW  ASSEMBLER   I2C INTERRUPT ROUTINE

LOC   OBJ          LINE  SOURCE

                      1  $TITLE (I2C INTERRUPT ROUTINE)
                      2  $PAGELENGTH(40)
                      3  ;
                      4          PUBLIC INT0_SRV
                      5          PUBLIC DIR,I2C_END
                      6          PUBLIC BASE,NR_BYTES,IIC_CNT,SLAVE
                      7          EXTRN  CODE(SENDBYTE,SENDCONTR,STOP)
                                 EXTRN  CODE(READBYTE,READCONTR)
                      8  ;
                      9  ;Define variables in RAM
                     10  IIC_VAR SEGMENT DATA
----                 11          RSEG IIC_VAR
0000:           R    12  BASE:   DS 1             ;Pointer to I2C table (till
                                                  ;256)
0001:                13  NR_BYTES: DS 1           ;Number of bytes to rcv/trm
0002:                14  IIC_CNT:DS 1             ;I2C byte counter
0003:                15  SLAVE:  DS 1             ;Slave address after START
                     16  ;
                     17  ;Define variable segment
                     18  BIT_VAR SEGMENT DATA BITADDRESSABLE
----                 19          RSEG BIT_VAR
0000:           R    20  STATUS: DS 1             ;Byte with flags
0000            R    21  I2C_END BIT STATUS.0     ;Defines if a I2C
                                                  ;transmission is finished
                     22                           ;'1' is finished
                     23                           ;'0' is not ready
0000            R    24  DIR     BIT STATUS.3     ;Defines direction of I2C
                                                  ;transmission
                     25                           ;'1':Transmit    '0':Receive
                     26  ;
                     27  ;Define code segment for routine
                     28  IIC_INT SEGMENT CODE PAGE
----                 29          RSEG IIC_INT
                     30  ;
                     31  ;Program uses registers in RB1
                     32          USING 1
                     33  ;
0000:           R    34  INT0_SRV:
0000: C0E0           35          PUSH ACC         ;Save acc. en psw on stack
0002: C0D0           36          PUSH PSW
0004: 75D008         37          MOV PSW,#08H     ;Select register bank 1
0007: 300016    R    38          JNB DIR,RECEIVE ;Test direction bit
                     39                           ;8584 is MST/TRM
                     40
                     41  ;Program part to transmit bytes to IIC bus
000A: E502      R    42          MOV A,IIC_CNT    ;Compare IIC_CNT and
                                                  ;NR_BYTES
000C: B50105    R    43          CJNE A,NR_BYTES,PROCEED
000F: 120000    R    44          CALL STOP        ;All bytes transmitted
0012: 8032           45          JMP EXIT
0014: A800      R    46  PROCEED:MOV R0,BASE      ;RAM pointer
0016: E6             47          MOV A,@R0        ;Source is internal RAM
0017: 0500      R    48          INC BASE         ;Update pointer of table
0019: 120000    R    49          CALL SENDBYTE    ;Send byte to IIC bus
001C: 0502      R    50          INC IIC_CNT      ;Update byte counter
001E: 8026           51          JMP EXIT
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
                    52  ;
                    53  ;
                    54  ;Program to receive byte from IIC bus
0020:               55  RECEIVE:
0020: E502      R   56          MOV A,IIC_CNT   ;Test if last byte is to be
                                                ;received
0022: 04            57          INC A
0023: 04            58          INC A
0024: B50105    R   59          CJNE A,NR_BYTES,PROC_RD
0027: 7448          60          MOV A,#01001000B;Last byte to be received.
                                                ;Disable ACK
0029: 120000    R   61          CALL SENDCONTR  ;Write control word to
                                                ;PCD8584
002C: 120000    R   62  PROC_RD:CALL READBYTE   ;Read I2C byte
002F: FC            63          MOV R4,A        ;Save accu
                    64  ;If RECEIVE is entered after the transmission of
                    65  ;START+address then the result of READBYTE is not
                    66  ;relevant. READBYTE is used to start the generation
                        ;of the clock pulses for the next byte to read.
                    67  ;This situation occurs when IIC_CNT is 0
0030: E4            68          CLR A           ;Test IIC_CNT
0031: B50202    R   69          CJNE A,IIC_CNT,SAVE
0034: 8006          70          JMP END_TEST    ;START is send. No relevant
                                                ;data in data reg. of 8584
0036: A800      R   71  SAVE:   MOV R0,BASE
0038: EC            72          MOV A,R4        ;Destination is internal RAM
0039: F6            73          MOV @R0,A
003A: 0500      R   74          INC BASE
003C: 0502      R   75  END_TEST:INC IIC_CNT    ;Test if all bytes are
                                                ;received
003E: E501      R   76          MOV A,NR_BYTES
0040: B50203    R   77          CJNE A,IIC_CNT,EXIT
0043: 120000    R   78          CALL STOP       ;All bytes received
                    79  ;
0046: D0D0          80  EXIT:   POP PSW         ;Restore PSW and accu
0048: D0E0          81          POP ACC
004A: 32            82          RETI
                    83  ;
004B:               84          END
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
ASM51  TSW  ASSEMBLER   Send a string of bytes to the PCF8577 on OM1016

LOC   OBJ          LINE  SOURCE

                      1  $TITLE (Send a string of bytes to the PCF8577 on
                             OM1016)
                      2  $PAGELENGTH(40)
                      3  ;
                      4  ;This program is an example to transmit bytes via
                         ;PCD8584
                      5  ;to the I2C-bus
                      6  ;
                      7          PUBLIC  SLAVE_ADR,I2C_CLOCK,PCD8584
                      8          EXTRN   CODE(I2C_INIT,INT0_SRV,START)
                      9          EXTRN   BIT(I2C_END,DIR)
                     10          EXTRN   DATA(BASE,NR_BYTES,IIC_CNT,SLAVE)
                     11  ;
                     12  ;
                     13  ;Define used segments
                     14  USER    SEGMENT CODE     ;Segment for user program
                     15  RAMTAB  SEGMENT DATA     ;Segment for table in
                                                  ;internal RAM
                     16  RAMVAR  SEGMENT DATA     ;Segment for RAM variables
                                                  ;in RAM
                     17  ;
                     18  ;
----                 19          RSEG    RAMVAR
0000:          R     20  STACK:  DS 20            ;Reserve stack area
                     21  ;
                     22  ;
----                 23          CSEG AT 00H
0000: 020000   R     24          JMP MAIN         ;Reset vector
                     25  ;
                     26  ;
----                 27          CSEG AT 03H
0003: 020000   R     28          JMP INT0_SRV     ;I2C interrupt vector
                                                  ;(INT0/)
                     29  ;
                     30  ;
----                 31          RSEG USER
                     32  ;Define I2C clock, own slave address and PCD8584
                         ;hardware address
0055                 33  SLAVE_ADR EQU 55H        ;Own slave address is 55H
001C                 34  I2C_CLOCK EQU 00011100B ;12.00MHz/90kHz
0000                 35  PCD8584   EQU 0000H      ;PCD8584 address with A0=0
                     36  ;0000: 7581FF   R    37  MAIN:   MOV SP,#STACK-1 ;Initialise stack pointer
                     38  ;Initialise 8031 interrupt registers for I2C
                         ;interrupt
0003: D2A8           39          SETB EX0         ;Enable interrupt INT0/
0005: D2AF           40          SETB EA          ;Set global enable
0007: D2B8           41          SETB PX0         ;Priority level '1'
0009: D288           42          SETB IT0         ;INT0/ on falling edge
                     43  ;
                     44  ;Initialise PCD8584
000B: 120000   R     45          CALL I2C_INIT
                     46  ;
                     47  ;Make a table in RAM with data to be transmitted.
000E: 120021   R     48          CALL MAKE_TAB
                     49  ;
                     50  ;Set variables to control PCD8584
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
0011: D200     R    51          SETB DIR          ;DIR='transmission'
0013: 750000   R    52          MOV BASE,#TABLE ;Start address of I2C-data
0016: 750005   R    53          MOV NR_BYTES,#05H ;5 bytes must be
                                                  ;transferred
0019: 750074   R    54          MOV SLAVE,#01110100B ;Slave address PCF8577
                                                  ; + WR/
001C: 120000   R    55          CALL START        ;Start I2C transmission
                    56  ;
                    57  ;
001F: 80FE          58  LOOP:   JMP LOOP          ;Endless loop when program
                                                  ;is finished
                    59  ;
                    60  ;
0021:               61  MAKE_TAB:
0021: 7800     R    62          MOV R0,#TABLE     ;Make data ready for I2C
                                                  ;transmission
0023: 7600          63          MOV @R0,#00       ;Controlword PCF8577
0025: 08            64          INC R0
0026: 76FC          65          MOV @R0,#0FCH     ;'0'
0028: 08            66          INC R0
0029: 7660          67          MOV @R0,#60H      ;'1'
002B: 08            68          INC R0
002C: 76DA          69          MOV @R0,#0DAH     ;'2'
002E: 08            70          INC R0
002F: 76F2          71          MOV @R0,#0F2H     ;'3'
0031: 22            72          RET
                    73  ;
                    74  ;
----                75          RSEG RAMTAB
0000:          R    76  TABLE:  DS 10             ;Reserve space in internal
                                                  ;data RAM
                    77                            ;for I2C data to transmit
                    78  ;
                    79  ;
000A:               80          END
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
ASM51  TSW  ASSEMBLER   Receive 2 bytes from the PCF8574P on OM1016

LOC   OBJ          LINE  SOURCE

                      1  $TITLE (Receive 2 bytes from the PCF8574P on OM1016)
                      2  $PAGELENGTH(40)
                      3  ;
                      4  ;This program is an example to receive bytes via
                         ;PCD8584
                      5  ;from the I2C-bus
                      6  ;
                      7          PUBLIC  SLAVE_ADR,I2C_CLOCK,PCD8584
                      8          EXTRN   CODE(I2C_INIT,INT0_SRV,START)
                      9          EXTRN   BIT(I2C_END,DIR)
                     10          EXTRN   DATA(BASE,NR_BYTES,IIC_CNT,SLAVE)
                     11  ;
                     12  ;
                     13  ;Define used segments
                     14  USER    SEGMENT CODE    ;Segment for user program
                     15  RAMTAB  SEGMENT DATA    ;Segment for table in
                                                 ;internal RAM
                     16  RAMVAR  SEGMENT DATA    ;Segment for RAM variables
                                                 ;in RAM
                     17  ;
                     18  ;
----                 19          RSEG    RAMVAR
0000:          R     20  STACK:  DS 20           ;Reserve stack area
                     21  ;
                     22  ;
----                 23          CSEG AT 00H
0000: 020000   R     24          JMP MAIN        ;Reset vector
                     25  ;
                     26  ;
----                 27          CSEG AT 03H
0003: 020000   R     28          JMP INT0_SRV    ;I2C interrupt vector
                                                 ;(INT0/)
                     29  ;
                     30  ;
----                 31          RSEG USER
                     32  ;Define I2C clock, own slave address and PCD8584
                         ;hardware address
0055                 33  SLAVE_ADR EQU 55H       ;Own slave address is 55H
001C                 34  I2C_CLOCK EQU 00011100B ;12.00MHz/90kHz
0000                 35  PCD8584   EQU 0000H     ;PCD8584 address with A0=0
                     36  ;0000: 7581FF   R    37  MAIN:   MOV SP,#STACK-1 ;Initialise stack pointer
                     38  ;Initialise 8031 interrupt registers for I2C
                         ;interrupt
0003: D2A8           39          SETB EX0        ;Enable interrupt INT0/
0005: D2AF           40          SETB EA         ;Set global enable
0007: D2B8           41          SETB PX0        ;Priority level '1'
0009: D288           42          SETB IT0        ;INT0/ on falling edge
                     43  ;
                     44  ;Initialise PCD8584
000B: 120000   R     45          CALL I2C_INIT
                     46  ;
                     47  ;Set variables to control PCD8584
000E: C200     R     48          CLR  DIR        ;DIR='receive'
0010: 750000   R     49          MOV BASE,#TABLE ;Start address of I2C-data
0013: 750002   R     50          MOV NR_BYTES,#02H ;2 bytes must be received
0016: 75004F   R     51          MOV SLAVE,#01001111B ;Slave address PCF8574
                                                 ; + RD
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers


```
0019: 120000   R    52          CALL START      ;Start I2C transmission
                    53  ;
                    54  ;
001C: 80FE          55  LOOP:   JMP LOOP        ;Endless loop when program
                                                ;is finished
                    56  ;
                    57  ;
----                58          RSEG RAMTAB
0000:          R    59  TABLE:  DS 10           ;Reserve space in internal
                                                ;data RAM
                    60                          ;for received I2C data
                    61  ;
                    62  ;
000A:               63          END
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

                      1  $TITLE (Demo program for PCD8584 I2C-routines)
                      2  $PAGELENGTH(40)
                      3  ;Program displays on the LCD display  the time (with
                         ;PCF8583). Dots on LCD display blink every second.
                      5  ;On the LED display the values of the successive
                         ;analog input channels are shown.
                      7  ;Program reads analog channels of PCF8591P.
                      8  ;Channel number and channel value are displayed
                         ;successively.
                      9  ;Values are displayed on LCD and LED display on I2C
                         ;demo board.
                     10  ;
                     11          PUBLIC    SLAVE_ADR,I2C_CLOCK,PCD8584
                     12          EXTRN     CODE(I2C_INIT,INT0_SRV,START)
                     13          EXTRN     BIT(I2C_END,DIR)
                     14          EXTRN     DATA(BASE,NR_BYTES,IIC_CNT,SLAVE)
                     15  ;
                     16  ;
                     17  ;Define used segments
                     18  USER    SEGMENT   CODE    ;Segment for user program
                     19  RAMTAB  SEGMENT   DATA    ;Segment for table in
                                                   ;internal RAM
                     20  RAMVAR  SEGMENT   DATA    ;Segment for variables
                     21  ;
----                 22          RSEG RAMVAR
0000:          R     23  STACK:  DS 20             ;Stack area (20 bytes)
0014:                24  PREVIOUS: DS 1           ;Store for previous seconds
0015:                25  CHANNEL:DS 1             ;Channel number to be
                                                   ;sampled
0016:                26  AN_VAL: DS 1             ;Analog value sampled
                                                   ;channel
0017:                27  CONVAL: DS 3             ;Converted BCD value sampled
                                                   ;channel
                     28  ;
----                 29          CSEG AT 00H
0000: 020000   R     30          LJMP MAIN        ;Reset vector
                     31  ;
----                 32          CSEG AT 03H      ;INT0/
0003: 020000   R     33          LJMP INT0_SRV    ;Vector I2C-interrupt
                     34  ;
                     35  ;
----                 36          RSEG USER37  ;Define I2C clock, own slave address and address for
                         ;main processor
0055                 38  SLAVE_ADR EQU 55H        ;Own slaveaddress is 55h
001C                 39  I2C_CLOCK EQU 00011100B ;12.00MHz/90kHz
0000                 40  PCD8584   EQU 0000H      ;Address of PCD8584. This
                                                   ;must be an EVEN number!!
                     41  ;Define addresses of I2C peripherals
00A3                 42  PCF8583R  EQU 10100011B ;Address PCF8583 with Read
                                                   ;active
00A2                 43  PCF8583W  EQU 10100010B ;Address PCF8583 with Write
                                                   ;active
009F                 44  PCF8591R  EQU 10011111B ;Address PCF8591 with Read
                                                   ;active
009E                 45  PCF8591W  EQU 10011110B ;Address PCF8591 with Write
                                                   ;active
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

0074                46  PCF8577W  EQU 01110100B ;Address PCF8577 with Write
                                                ;active
0076                47  SAA1064W  EQU 01110110B ;Address SAA1064 with Write
                                                ;active
                    48  ;
0000: 7581FF   R    49  MAIN:   MOV SP,#STACK-1 ;Define stack pointer
                    50  ;Initialise 80C31 interruptregisters for I2C
                        ;interrupt (INT0/)
0003: D2A8          51          SETB EX0        ;Enable interrupt INT0/
0005: D2AF          52          SETB EA         ;Set global enable
0007: D2B8          53          SETB PX0        ;Priority level is '1'
0009: D288          54          SETB IT0        ;INT0/ on falling edge
                    55  ;Initialise PCD8584
000B: 120000   R    56          CALL I2C_INIT
                    57  ;
000E: 751500   R    58          MOV CHANNEL,#00 ;Set AD-channel
                    59  ;
                    60  ;Time must be read from PCD8583.
                    61  ;First write word address and control register of
                        ;PCD8583.
0011: D200     R    62          SETB DIR        ;DIR='transmission'
0013: 750000   R    63          MOV BASE,#TABLE ;Start address I2C data
0016: 750002   R    64          MOV NR_BYTES,#02H ;Send 2 bytes
0019: 7500A2   R    65          MOV SLAVE,#PCF8583W
001C: E4            66          CLR A
001D: F500     R    67          MOV TABLE,A     ;Data to be sent (word
                                                ;address).
001F: F501     R    68          MOV TABLE+1,A   ;      "         (control
                                                ;byte)
0021: 120000   R    69          CALL START      ;Start transmission.
0024: 3000FD   R    70  FIN_1:  JNB I2C_END,FIN_1 ;Wait till transmission
                                                ;finished
                    71  ;Send word address before reading time
0027: D200     R    72  REPEAT: SETB DIR        ;'transmission
0029: 750000   R    73          MOV BASE,#TABLE ;I2C data
002C: 7500A2   R    74          MOV SLAVE,#PCF8583W
002F: 7401          75          MOV A,#01
0031: F500     R    76          MOV NR_BYTES,A  ;Send 1 byte
0033: F500     R    77          MOV TABLE,A     ;Data to be sent is '1'
0035: 120000   R    78          CALL START      ;Start I2C transmission
0038: 3000FD   R    79  FIN_2:  JNB I2C_END,FIN_2 ;Wait till transmission
                                                ;finished
                    80  ;
                    81  ;Time can now be read from PCD8583. Data read is
                    82  ;hundredths of sec's, sec's, min's and hr's
003B: C200     R    83          CLR DIR         ;DIR='receive'
003D: 750000   R    84          MOV BASE,#TABLE ;I2C table
0040: 750004   R    85          MOV NR_BYTES,#04; 4 bytes to receive
0043: 7500A3   R    86          MOV SLAVE,#PCF8583R
0046: 120000   R    87          CALL START      ;Start I2C reception
0049: 3000FD   R    88  FIN_3:  JNB I2C_END,FIN_3 ;Wait till finished
                    89  ;
                    90  ;Transfer data to R2...R5
004C: 7800     R    91          MOV R0,#TABLE   ;Set pointers
004E: 7902          92          MOV R1,#02H     ;Pointer R2
0050: E6            93  TRANSFER:MOV A,@R0
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

0051: F7            94          MOV @R1,A
0052: 08            95          INC R0
0053: 09            96          INC R1
0054: D500F9   R    97          DJNZ NR_BYTES,TRANSFER
0057: ED            98          MOV A,R5        ;Mask of hour counter
0058: 543F          99          ANL A,#3FH
005A: FD           100          MOV R5,A
                   101  ;
                   102  ;Data must now be displayed on LCD display.
                   103  ;First minutes and hours (in R4 and R5) must be
                   104  ;converted from BCD to LCD segment data.The segment
                        ;data
                   105  ;will be transferred to TABLE. R0 is pointer to
                        ;table
005B: 7800     R   106          MOV R0,#TABLE
005D: 7600         107          MOV @R0,#00H    ;Control word for PCF8577
005F: 08           108          INC R0 0060: 120080   R   109          CALL CONV
                   110  ;
                   111  ;Switch on dp between hours and minutes
0063: 430301   R   112          ORL TABLE+3,#01H
                   113  ;If lsb of seconds is '0' then switch on dp.
0066: EB           114          MOV A,R3        ;Get seconds
0067: 13           115          RRC A           ;lsb in carry
0068: 4003         116          JC PROCEED
006A: 430101   R   117          ORL TABLE+1,#01H;switch on dp
                   118  ;
                   119  ;Now the time (hours,minutes) can be displayed on
                        ;the LCD
006D:              120  PROCEED:
006D: D200     R   121          SETB DIR        ;Direction 'transmit'
006F: 750000   R   122          MOV BASE,#TABLE
0072: 750005   R   123          MOV NR_BYTES,#05H
0075: 750074   R   124          MOV SLAVE,#PCF8577W
0078: 120000   R   125          CALL START      ;Start transmission
                   126  ;
007B: 3000FD   R   127  FIN_4:  JNB I2C_END,FIN_4
007E: 8026         128          JMP ADCON       ;Proceed with AD-conversion
                                                ;part
                   129  ;
                   130 ;*****************************************************************
                   131  ;Routines used by clock part of demo
                   132  ;
                   133  ;CONV converts hour and minute data to LCD data and
                        ;stores
                   134  ;it in TABLE.
0080: 90009C   R   135  CONV:   MOV DPTR,#LCD_TAB ;Base for LCD segment
                                                ;table
0083: ED           136          MOV A,R5        ;Hours to accu
0084: C4           137          SWAP A          ;Swap nibbles
0085: 120096   R   138          CALL LCD_DATA   ;Convert 10's hours to LCD
                                                ;data in table
0088: ED           139          MOV A,R5        ;Get hours
0089: 120096   R   140          CALL LCD_DATA
008C: EC           141          MOV A,R4        ;Get minutes
008D: C4           142          SWAP A
008E: 120096   R   143          CALL LCD_DATA   ;Convert 10's minutes
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

0091: EC           144          MOV A,R4
0092: 120096   R   145          CALL LCD_DATA   ;Convert minutes
0095: 22           146          RET
                   147  ;
                   148  ;LCD_DATA gets data from segment table and stores it
                        ;in TABLE
0096: 540F         149  LCD_DATA:ANL A,#0FH     ;Mask off LS-nibble
0098: 93           150          MOVC A,@A+DPTR  ;Get LCD segment data
0099: F6           151          MOV @R0,A       ;Save data in table
009A: 08           152          INC R0
009B: 22           153          RET
                   154  ;
                   155  ;LCD_TAB is conversion table for LCD
009C:              156  LCD_TAB:
009C: FC60DA       157          DB 0FCH,60H,0DAH; '0','1','2'
009F: F266B6       158          DB 0F2H,66H,0B6H; '3','4','5'
00A2: 3EE0FE       159          DB 3EH,0E0H,0FEH; '6','7','8'
00A5: E6           160          DB 0E6H         ; '9'
                   161  ;
                   162 ;*************************************************************************
                   163  ;
                   164  ;
                   165  ;These part of the program reads an analog
                        ;input-channel.
                   166  ;Displaying is done on the LED-display
                   167  ;On odd-seconds the channel number will be
                        ;displayed.
                   168  ;On even-seconds the analog value of this channel is
                        ;displayed
                   169  ;Then the next channel is displayed.
                   170  ;
00A6: EB           171  ADCON:  MOV A,R3         ;Get seconds
00A7: 13           172          RRC A           ;lsb to carry
00A8: 503C         173          JNC NEW_MEAS    ;Even seconds; do a
                                                ;measurement on the current
                                                ;channel
                   174  ;
                   175  ;Display and/or update channel
00AA: 33           176          RLC A           ;Restore accu
00AB: B51402   R   177          CJNE A,PREVIOUS,NEW_CH ;If new seconds,
                                                ;update channel number
00AE: 800A         178          JMP DISP_CH
00B0: 0515     R   179  NEW_CH: INC CHANNEL
00B2: E515     R   180          MOV A,CHANNEL   ;If channel=4 then
                                                ;channel:=0
00B4: B40403       181          CJNE A,#04,DISP_CH
00B7: 751500   R   182          MOV CHANNEL,#00
00BA: 8B14     R   183  DISP_CH:MOV PREVIOUS,R3 ;Update previous seconds
00BC: E515     R   184          MOV A,CHANNEL   ;Get segment value of
                                                ;channel
00BE: 900193   R   185          MOV DPTR,#LED_TAB
00C1: 93           186          MOVC A,@A+DPTR
                   187  ;
00C2: 7800     R   188          MOV R0,#TABLE   ;Fill table with I2C data
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

00C4: 7600         189          MOV @R0,#00     ;SAA1064 instruction byte
00C6: 08           190          INC R0
00C7: 7677         191          MOV @R0,#77H    ;SAA1064 control byte
00C9: 08           192          INC R0
00CA: F6           193          MOV @R0,A       ;Channel number
00CB: E4           194          CLR A
00CC: 08           195          INC R0
00CD: F6           196          MOV @R0,A       ;Second digit
00CE: 08           197          INC R0
00CF: F6           198          MOV @R0,A       ;Third digit
00D0: 08           199          INC R0
00D1: F6           200          MOV @R0,A       ;Fourth byte
                   201  ;
00D2: D200     R   202          SETB DIR        ;I2C transmission of channel
                                                ;number
00D4: 750000   R   203          MOV BASE,#TABLE
00D7: 750006   R   204          MOV NR_BYTES,#06H
00DA: 750076   R   205          MOV SLAVE,#SAA1064W
00DD: 120000   R   206          CALL START
                   207  ;
00E0: 3000FD   R   208  FIN_5:  JNB I2C_END,FIN_5
00E3: 020027   R   209          JMP REPEAT      ; Repeat clock and AD cycle
                                                ; again
                   210  ;
                   211  ;
                   212  ;Measure and display the value of an AD-channel
00E6: 120108   R   213  NEW_MEAS: CALL AD_VAL   ;Do measurement
                   214  ;Wait till values are available
00E9: 3000FD   R   215  FIN_6:  JNB I2C_END,FIN_6
                   216  ;Relevant byte in TABLE+1. Transfer to AN_VAL
00EC: 7801     R   217          MOV R0,#TABLE+1
00EE: 8616     R   218          MOV AN_VAL,@R0
00F0: E516     R   219          MOV A,AN_VAL    ;Channel value in accu for
                                                ;conversion
                   220  ;AN_VAL is converted to BCD value of the measured
                        ;voltage.
                   221  ;Input value for CONVERT in accu
                   222  ;Address for MSByte in R1
00F2: 7917     R   223          MOV R1,#CONVAL
00F4: 120154   R   224          CALL CONVERT
                   225  ;Convert 3 bytes of CONVAL to LED-segments
00F7: 900193   R   226          MOV DPTR,#LED_TAB ;Base of segment table
00FA: 7817     R   227          MOV R0,#CONVAL
00FC: 12018A   R   228          CALL SEG_LOOP
                   229  ;Display value of channel to LED display
00FF: 12012C   R   230          CALL LED_DISP
0102: 3000FD   R   231  FIN_8:  JNB I2C_END,FIN_8 ;Wait till I2C
                                                ;transmission is ended
0105: 020027   R   232          JMP REPEAT      ;Repeat clock and AD cycle
                   233  ;
                   234 ;*************************************************************
                   235  ;Routines used for AD converter.
                   236  ;
                   237  ;AIN reads an analog values from channel denoted by
                        ;CHANNEL.
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

                    238  ;Send controlbyte:
0108: D200      R   239  AD_VAL: SETB DIR        ;I2C transmission
010A: 7800      R   240          MOV R0,#TABLE   ;Define control word
010C: A615      R   241          MOV @R0,CHANNEL
010E: 750000    R   242          MOV BASE,#TABLE ;Set base at table
0111: 750001    R   243          MOV NR_BYTES,#01H ;Number of bytes to be
                                                   ;send
0114: 75009E    R   244          MOV SLAVE,#PCF8591W ;Slave address PCF8591
0117: 120000    R   245          CALL START      ;Start transmission of
                                                   ;controlword
011A: 3000FD    R   246  FIN_7:  JNB I2C_END,FIN_7 ;Wait until tranmission is
                                                   ;finished
                    247  ;Read 2 data bytes from AD-converter
                    248  ;First data byte is from previous conversion and not
                    249  ;relevant
011D: C200      R   250          CLR DIR         ;I2C reception
011F: 750000    R   251          MOV BASE,#TABLE ;Bytes must be stored in
                                                   ;TABLE
0122: 750002    R   252          MOV NR_BYTES,#02H; Receive 3 bytes
0125: 75009F    R   253          MOV SLAVE,#PCF8591R ;Slave address PCF8591
0128: 120000    R   254          CALL START
012B: 22            255          RET
                    256  ;
                    257  ;LED_DISP displays the data of 3 bytes from address
                         ;CONVAL
012C:               258  LED_DISP:
012C: 431780    R   259          ORL CONVAL,#80H ;Set decimal point
012F: 7800      R   260          MOV R0,#TABLE
0131: 7917      R   261          MOV R1,#CONVAL
0133: 7600          262          MOV @R0,#00     ;SAA1064 instruction byte
0135: 08            263          INC R0
0136: 7677          264          MOV @R0,#01110111B ;SAA1064 control byte
0138: 08            265          INC R0
0139: 7600          266          MOV @R0,#00     ;First LED digit
013B: 08            267          INC R0
013C: 120185    R   268          CALL GETBY      ;Second digit
013F: 120185    R   269          CALL GETBY      ;Third digit
0142: 120185    R   270          CALL GETBY      ;Fourth digit
0145: D200      R   271          SETB DIR        ;I2C transmission
0147: 750000    R   272          MOV BASE,#TABLE
014A: 750006    R   273          MOV NR_BYTES,#06
014D: 750076    R   274          MOV SLAVE,#01110110B
0150: 120000    R   275          CALL START      ;Start I2C transmission
0153: 22            276          RET
                    277  ;
                    278  ;CONVERT calculates the voltage of the analog value.
                    279  ;Analog value must be in accu
                    280  ;BCD result (3 bytes) is stored from address stored
                         ;in R1
                    281  ;Calculation: AN_VAL*(5/256)
0154: 75F005        282  CONVERT:MOV B,#05
0157: A4            283          MUL AB
                    284  ;b2..b0 of reg. B : 2E+2..2E0
                    285  ;b7..b0 of accu   : 2E-1..2E-8
0158: A7F0          286          MOV @R1,B       ;Store MSB (10E0-units)
015A: 09            287          INC R1
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

015B: 7700         288          MOV @R1,#00      ;Calculate 10E-1 unit
                                                 ;(10E-1 is 19h)
015D: B41C02       289  TEN_CH: CJNE A,#19H+03H,V1  ;Check if accu <= 0.11
0160: 8002         290          JMP TENS         ;accu=0.11; update tens
0162: 4006         291  V1:     JC NX_CON        ;accu<0.11; update hundreds
0164: C3           292  TENS:   CLR C            ;Calculate new value
0165: 9419         293          SUBB A,#19H
0167: 07           294          INC @R1          ;Update BCD byte
0168: 80F3         295          JMP TEN_CH
                   296  ;Correction may be neccessary. With 8 bits '0.1' is
                        ;in fact 0.0976.
                   297  ;A digit of '0A' may appear. Correct this by
                        ;decrementing the digit.
                   298  ;The intermediate result result must be corrected
                        ;with 10*(0.1-0.0976)
                   299  ;This is 06H
016A: B70A03       300  NX_CON: CJNE @R1,#0AH,PROC_CON ; If digit is '0A'
                                                 ;then correct
016D: 17           301          DEC @R1
016E: 2419         302          ADD A,#19H
0170: 09           303  PROC_CON:INC R1
0171: 7700         304          MOV @R1,#00      ;Calculate 10E-2 units
0173: B40302       305  HUND:   CJNE A,#03H,V2  ;Check if accu <= 10E-2
0176: 8002         306          JMP HUNS         ;accu=10E-2; update hundreds
0178: 4006         307  V2:     JC FINISH        ;accu<10E-2; conversion
                                                 ;finished
017A: C3           308  HUNS:   CLR C            ;Calculate new value
017B: 9403         309          SUBB A,#03H
017D: 07           310          INC @R1          ;Update BCD byte
017E: 80F3         311          JMP HUND
0180: B70A01       312  FINISH: CJNE @R1,#0AH,FIN ;Check if result is '0A'.
                                                 ;Then correct.
0183: 17           313          DEC @R1
0184: 22           314  FIN:    RET
                   315  ;
                   316  ;CALLBY tranfers byte from @R1 to @R0
0185: E7           317  GETBY:  MOV A,@R1
0186: F6           318          MOV @R0,A
0187: 08           319          INC R0
0188: 09           320          INC R1
0189: 22           321          RET
                   322  ;
                   323  ;SEG_LOOP converts 3 values to segment values.
                   324  ;R0 contains address of source and destination
                   325  ;DPTR contains base of table
018A: 7903         326  SEG_LOOP: MOV R1,#03    ;Loop counter
018C: E6           327  INLOOP: MOV A,@R0     ;Get value to be displayed
018D: 93           328          MOVC A,@A+DPTR  ;Get segment value from
                                                 ;table
018E: F6           329          MOV @R0,A       ;Store segment data
018F: 08           330          INC R0
0190: D9FA         331          DJNZ R1,INLOOP
0192: 22           332          RET
                   333  ;
                   334  ;
```

Interfacing the PCD8584 I²C-bus controller to 80C51 family microcontrollers

AN425

```
ASM51  TSW  ASSEMBLER   Demo program for PCD8584 I2C-routines

LOC   OBJ          LINE  SOURCE

                   335  ;LED_TAB is conversion table for BCD to LED segments
0193:              336  LED_TAB:
0193: 7D483E       337          DB 7DH,48H,3EH  ; '0','1','2'
0196: 6E4B67       338          DB 6EH,4BH,67H  ; '3','4','5'
0199: 734C7F       339          DB 73H,4CH,7FH  ; '6','7','8'
019C: 4F           340          DB 4FH          ; '9'
                   341  ;
                   342 ;*************************************************************
                   343  ;
----               344          RSEG RAMTAB
0000:          R   345  TABLE:  DS 10
                   346  ;
000A:              347          END
```

Controlling aire core meters with the 87C751 and SA 5775

AN426

Author: Bill Houghton

INTRODUCTION

Often, certain classes of microcontroller applications surface where large amounts of on-chip resources such as a large program memory space and numerous I/O pins are not required. These applications are typically cost sensitive and desirable attributes of the MCU include low cost and modest on-chip resources such as program and data memory, I/O, and timer-counters. Substaintial benefits of reduced design cycle time can be realized by using an industry-standard architecture having software compatibility with existing popular microcontrollers.

THE 87C751

The Philips 87C751 is one such microcontroller that easily meets these requirements. This device, shown in Figure 1, has a 2k x 8 program memory, 64 bytes of RAM, 19 parallel I/O lines, and a 16-bit autoreload timer-counter. It also includes an I^2C serial interface and a fixed rate timer. The 87C751 is based on the 80C51 core and thus uses an industry-standard architecture and instruction set. The device is available in both ROM (83C751) and EPROM (87C751) versions. The EPROM version is available in both UV erasable and OTP packages. References to the 87C751 in this document also apply to the 83C751, unless explicitly stated.

TYPICIAL APPLICATION

A typicial example of such an application is the interface between the 87C751 and the Philips SA 5775 Air Core Meter Driver shown in Figure 2. This circuit includes the 87C751 microcontroller, the SA 5775 air core meter driver, an NE555 timer, and discrete support components.

An air core meter differs from a conventional (d'Arsonval) meter movement in that it has no spring to return the needle to a predetermined position, no zeroing adjustment, and no permanent magnet in the classical sense. Instead, it consists of two coils of wire wound in quadrature with each other around a central core in which there is a disc magnetized along its diameter. A shaft is placed through the center of this disc so that the shaft rotates with the disc. An indicating needle attached to this shaft will rotate with it.

SA 5775 Air Core Meter Driver

The SA 5775 is a monolithic driver for controlling air core meters typically used in automotive instrument clusters and is shown in Figure 3. The SA 5775 receives a 10-bit serial word and converts that word to four voltage outputs that appear at the SINE+, SINE–, COSINE+, and COSINE– outputs. The differential voltage at the SINE outputs are applied to one coil of the meter and the COSINE outputs are applied to the other coil of the meter.

The currents through these coils produce a resultant magnetic force which is the vector sum of the magnetic forces produced by each of the two coils. Since the currents through the coils are bidirectional this magnetic vector can rotate through a full 360 degrees. The magnetized disc within the air core meter will follow the rotating vector and the needle will indicate the vector's current position. Since 10 bits are used, there are 1024 discrete words available resulting in an angular displacement of 0.3516 degrees per bit. This is small enough to provide an apparently smooth movement of the needle. The smoothness of the motion will depend greatly on the damping factor of the meter movement.

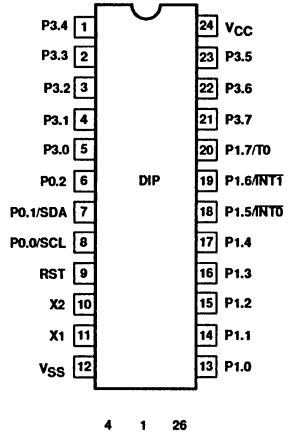
A simplified block diagram of the SA5775 is shown in Figure 4. This device consists of a serial-in/parallel-out shift register, a data latch, a D/A converter, buffers, and an internal voltage reference.

A logic high must be present on the chip select (CS) input to clock in the data. Data appearing on the data input (DI) pin is clocked into the shift register on the rising edge of the clock (CLK) input. The data output (DO) pin is the overflow from the shift register, allowing the user to daisy chain multiple SA5775 devices. Note that data is clocked out of this pin on the falling edge of the clock. The CS pin is also used to latch the parallel outputs of the shift register into the data latch. The outputs of the data latch feed the inputs to the D/A converter. The D/A converter outputs are buffered to form the drive signals for the meter coils.

A voltage reference for the D/A converter is provided internally. It is possible to externally force different values for these voltages and pins are provided for this purpose. However, this is not generally recommended as this could lead to increased power dissipation.

The D/A converter circuits and its associated output buffers are purposely designed such that the span of these circuits does not include the power supply rails. This is to avoid inaccuracies that would otherwise occur if the output were to become very close to either supply rail. With a supply voltage of 14 volts (VIGN), the positive reference (VREF+) is approximately 8 volts and the negative reference (VREF–) is approximately 1 volt. The outputs will then span a range from 1 volt to approximately 11 volts. The maximum output is [(VREF+) + 0.41((VREF+)–(VREF–))]. The SA5775 is designed to drive air core meters having a minimum winding impedance of 200 ohms.

The clock high and low time requirements are each 200 ns minimum, implying a maximum data rate approaching 2.5 megabits per second. At this rate it would require approximately 4 ms to ramp from zero to full scale if all binary codes were loaded into the SA5775. However, the air core meter cannot respond to such data rates.



Pin	Function	Pin	Function
1	P3.4	15	P1.0
2	P3.3	16	P1.1
3	P3.2	17	P1.2
4	P3.1	18	P1.3
5	N.C.	19	P1.4
6	P3.0	20	P1.5/INT0
7	P0.2	21	N.C.
8	P0.1/SDA	22	N.C.
9	P0.0/SCL	23	P1.6/INT1
10	N.C.	24	P1.7/T0
11	RST	25	P3.7
12	X2	26	P3.6
13	X1	27	P3.5
14	VSS	28	VCC

Figure 1. Pin Configuration

Controlling aire core meters with the 87C751 and SA 5775

AN426

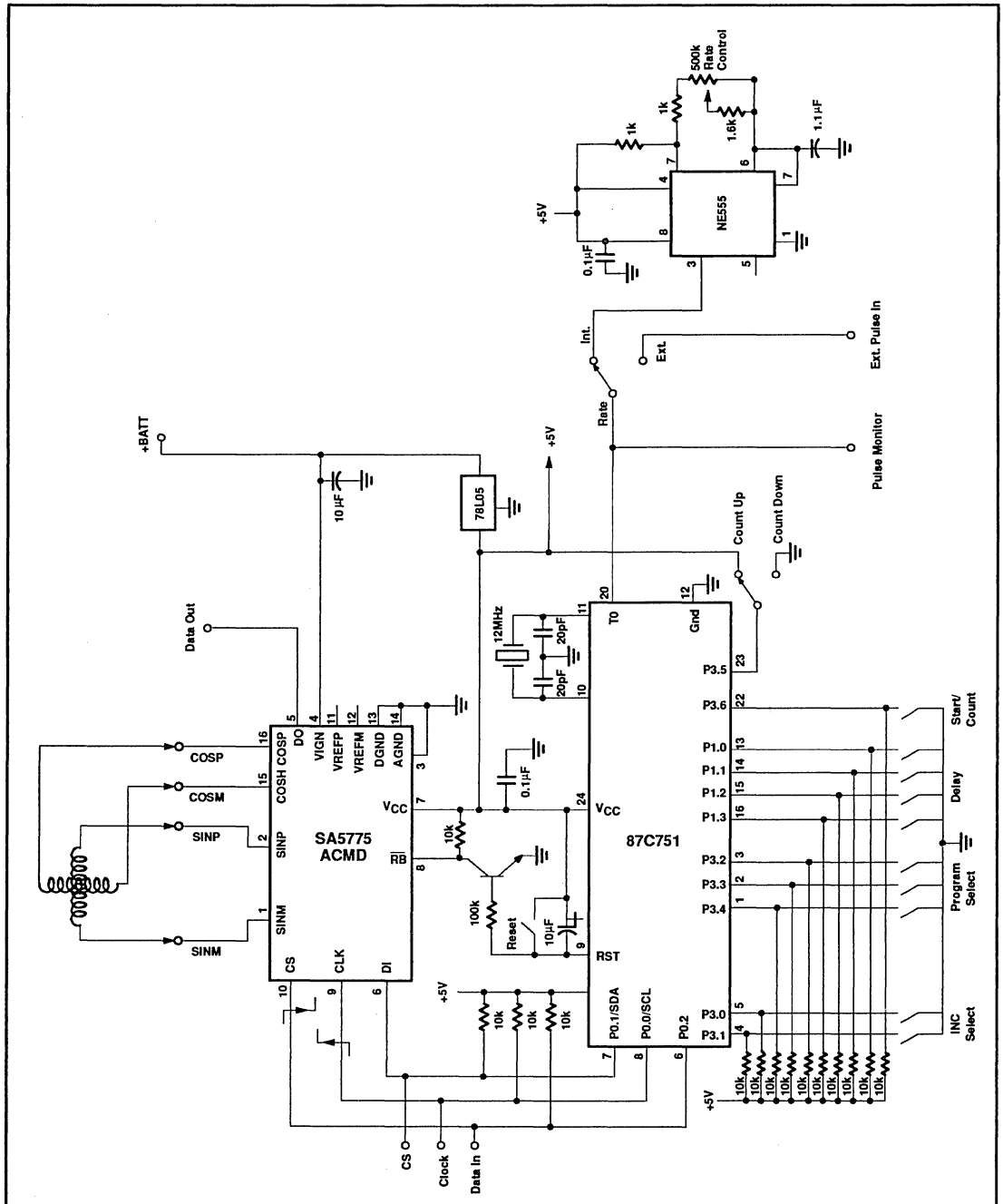
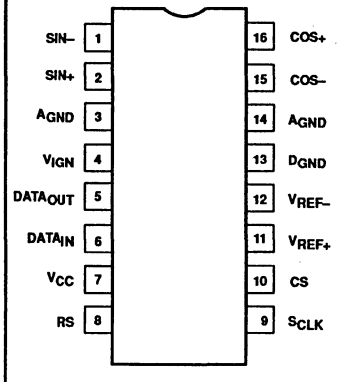


Figure 2. Interface Between the 87C751 and the Philips SA 5775

Controlling aire core meters with the 87C751 and SA 5775

AN426



NOTE: D package available in large SO (SOL) package only.

Figure 3. D and N Packages

87C751 Microcontroller

The 87C751 microcontroller provides all of the intelligence in this application. It samples various input ports to determine which demonstration programs to run, the incremental step sizes for angular displacement of the meter core, and the time delay between increments. In one of the demonstration modes, it also samples a variable frequency input and positions the meter core in response to the frequency of that input. The 87C751 also transmits the 10-bit serial data to the SA5775. Data input (DI), Clock (CLK), and Chip Select (CS) lines are driven from the 87C751.

Port 0 of the 87C751 is a 3-bit wide port and is used for communicating data to the ACMD. Data is transmitted, MSB first, in a serial stream clocked into the DI of the SA5775 on the rising edge of the clock. In order to clock in data, the CS pin of the SA5775 must be high. The data in the input register is shifted into a latch that drives the DAC on the high to low transition of the CS line. As data is shifted into the ACMD, it overflows through the Data Out (DO) pin on the falling edge of the clock. With this facility, multiple ACMDs can be daisy-drained with DO of one ACMD being connected to DI of the next one, and common clock and chip select lines may be used. This simplifies the interfacing to multiple meter drivers.

The 78L05 regulator (Q2) provides 5 Volt power for the board so that single supply of +14 volts can be applied to the board.

Three rotary switches are used on this board. The PROGRAM SELECT switch (S3) is used to select the program routine that is executed, the INC SELECT (S2) switch selects the incremental step sizes of the two of the routines, and the DELAY switch (S4) is used to set the delay between successive word transmissions in one of the routines.

The START/COUNT button (S5) is used to begin execution of a routine, and to cause the next incremental step in Routine #1.

The COUNT UP/DOWN switch (S6) is used in Routine #1 to determine whether the count is increased or decreased with transmission of successive words.

NE555 Timer

The NE555 timer shown in this application example is used as a free running squarewave generator used to simulate sensor inputs such as those which might be found in an automobile, etc. The NE555 timer (U4) operates in the astable mode to produce an output frequency that can be varied from about 1Hz to about 200 Hz. Three of the program routines measure the input period and produce an output code that is proportional to the frequency present at pin 20 (TO) of the microcontroller. A RATE switch (S7) is used to select between the on board oscillator or an external source.

The program listing is included at the end of this application note.

Program Entry

The program starts at address 030(hex) on line 21 of the program listing. The first task is to write 1's to all pins of each port.

Lines 25 and 26 clear registers 6 and 7. These registers are used in this program only to hold the data that is sent out to the ACMD. The registers are cleared to be sure that the starting value is zero.

At line 27 the program waits until the START/COUNT button (S5) is depressed before continuing. Lines 28 and 29 set the timer to overflow after 10ms. This is done by setting the timer registers for a count of 10,000 microseconds less than full scale. When the timer counter overflows the timer flag is set, and the timer is reloaded with the value in the timer register. By examining the timer flag we know when 10ms has expired.

Line 30 calls subroutine RPS (Read Port Selected), which reads Port 3 to determine which routine has been selected. Since the PROGRAM SELECT switch (S3) is connected to port pins P3.2 through P3.4, subroutine RPS (lines 507 through 511 at the end of the program) first reads Port 3 into the accumulator, then complements it because the switches used are complementary binary. The reading is then rotated right once and the upper nibble and the LSB (least significant bit) are masked off, leaving twice the value of the port selected in the accumulator. Twice the read value is needed for the next few main program lines that determine which routine to execute.

Line 31 moves the address of label JMPTBL (Jump Table) to the 16-bit Data Pointer (DPTR) register. Line 32 causes a program jump to the address that is the sum of the value in the accumulator (two times the routine number selected) plus the DPTR register. Since each of the commands on lines 33 through 40 are two byte commands, these addresses are all separated by two bytes; hence, the need for the accumulator to contain a number that is twice the number of the selected routine.

Routine 0

This routine begins on line 41 by incrementing the 10-bit word in registers 7 and 6 by the amount indicated by the setting of the INCREMENT SELECT switch, then sending that word to the SA5775. When a full scale overflow is detected, a full scale code (3FF hex) is sent out, followed by a delay of 500 ms, then successive output codes are sent out, decremented by an amount indicated by the INCREMENT SELECT switch. When an underflow is detected a code of zero scale is sent and the routine returns to the beginning of the program. This routine is implemented with a series of subroutine calls.

The SO subroutine begins on line 356 and starts by sending out whatever ten bits that in the two LSBs of register 7 (R7) plus the 8 bits of R6 by calling the SENDIT subroutine. Then it calls the UP subroutine, which increases the word value to be sent out. The program then jumps to the beginning of this subroutine, repeating the process of sending out a word and incrementing to the next word until an overflow from the tenth bit (bit 2 of R7) is detected at line 362.

The SENDIT subroutine (beginning on line 476) brings the CS line high, sets a bit counter (R1) to 2 (to send out two bits of R7), brings the value of R7 to the accumulator, rotates the accumulator to the right three times through the carry bit to bring the two LSBs to the position of the two MSBs, calls the SEND1 routine, which sends the number of bits in the accumulator, starting with the MSB, indicated by R1. Counter R1 is then set to 8 to send out all 8 bit of R6 and the accumulator is loaded with the contents of R6. The SEND1 routine is again called to send out the final 8 bits, and, on line 491, the CS line is brought low, loading the SA5775 internal parallel latch with the contents of the input shift register.

Controlling aire core meters with the 87C751 and SA 5775

AN426

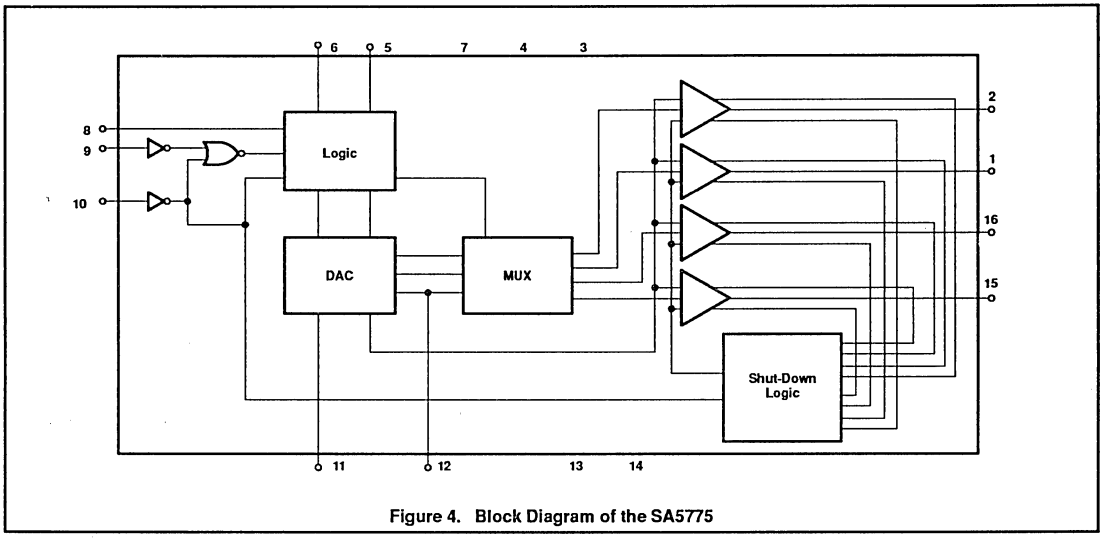


Figure 4. Block Diagram of the SA5775

The SEND1 routine rotates the accumulator left through the carry bit, moves the value of the carry bit to port pin PO.1 (SDA – Serial Data pin), waits to provide a setup time, brings the clock low, waits, brings the clock high, waits, then decrements bit counter sends the next bit if the counter is not zero. A return is executed when the counter becomes zero.

The UP subroutine, beginning at line 364, reads the delay selected by switch S4 at port pin P1, complements it (again, because the rotary switches are complementary binary), masks off the upper four bits (because the delay switch has just four positions and is connected to the lower four bits of the port), multiplies it by 4 (rotates left twice), then moves the result to R1. If R1 is not zero, the program jumps around line 376 and calls a 10ms delay (subroutine DLY10MS) the number of times entered into R1.

The 10ms delay subroutine (starting at line 436) sets the timer for 10ms, waits at line 446 for the timer flag to be set, clears the timer flag, stops the timer, and returns, in this case, to line 379, where the program decrements R1 and repeats the 10ms delay until R1 is zero.

If the selected delay was zero, the program jumps from line 376 to line 380 and reads port 3 to determine the amount the sent out word is to change from the value previously sent out. The accumulator is complemented and the upper 6 bits masked off to recover only the two bits of the selected increment amount. Since increments of 1, 2, 3, or 4 LSBs are hardly noticeable, the program then multiplies the result by 8 (rotate left three times). To insure a minimum change amount, the accumulator is increment by one at line 386. This all means that the increment amounts that can be selected are 1, 9, 17, or 25 LSBs. This amount is added, in lines 387 through 391, to the word previously send out and we return from this subroutine.

After calling the S0 subroutine, PR0GO call the FULLSC (full scale) subroutine, which sends out the full scale code of 3E8(hex). Although a 10-bit full scale code would be 3FF(hex), going only to 3E8 allows an easy distinction between zero scale and full scale when looking at the display. The FULLSC subroutine is found at line 352.

After advancing to full scale, there is a 500ms delay, found at line 464 and called from line 48, then 49 calls the S0D subroutine to send out decreasing word values.

The S0D subroutine begins at line 393 and begins by sending out the current word in R7 and R6 from line 398, then calling subroutine DOWN, which calculates the next (decreasing) word to send out. DOWN begins at line 402. It essentially does the same thing as the UP subroutine, but subtracts the INCREMENT SELECT value from the previously sent word rather than adding to it.

At line 50 subroutine ZEROSC is called to send a zero scale code to the SA5775, then the program branches back to the beginning.

Routine 1

This routine is selected with the PROGRAM SELECT switch is in position 1 or position 9. Routine 1 (PROG1) increments or decrements the word send out, depending upon the setting of the COUNT UP/COUNT DOWN switch, S6. The amount of change is determined by the setting of the INC SELECT switch, S2.

At line 63, the program examines S6 at port pin P3.6 and jumps to the decrement portion of the routine if the pin is low. If this pin is high, the UP subroutine is called from line 64 to increase the R7/R6 word value. The UP subroutine was previously described.

If pin P3.6 is low, the DOWN subroutine (line 402) decreases the previous word sent out by the amount determined from the INC SELECT switch setting.

To insure enough delay to allow the user time to release the START/COUNT button (S5), a delay of 200ms is included at line 66 before jumping to line 27, where another depression of the START/COUNT button is awaited. If S3 (PROGRAM SELECT) is still set to 1 or 9, depression of S5 will cause a jump back to line 52. If another program is selected, the program will jump to the selected routine.

Holding down S5 with PROGRAM SELECT set at position 1 or 9 will cause increasing or decreasing word values to be sent to the SA5775.

Controlling aire core meters with the 87C751 and SA 5775

AN426

Routine 2
PROG2 is the most complex of all these routines. The purpose of this routine is to cause the air core meter deflection to represent the frequency presented at the timer/counter input to the microcontroller. This is done by measuring the period of the input square wave and taking the inverse of the period. The input here must be a square wave because a slow rise and fall time at this input will cause fluctuating readings. To determine the frequency by counting pulses for a time would require a much longer time and, therefore, is impractical.

The MEAS (measure) subroutine is called at line 79 to measure the period of the input waveform and the CALC (calculate) subroutine is called at line 80 to calculate the code to send to the SA5775. The SENDIT subroutine is then called to send the word to the SA5775 and the program jumps back to line 28.

The MEAS subroutine begins at line 83 by being sure the timer is not running and clearing the timer (overflow) flag, then entering zero into both high and low bytes of the timer and the timer register. The carry bit is then cleared (line 90) and the timer started and the timer interrupt enabled.

Lines 93 and 94 form a short loop that waits until either the carry bit is set or until the TO input is low. The carry bit is set when the timer has gone beyond one second. This is done by the timer interrupt subroutine, found at lines 16 through 19. If the TO input never goes low, we know the frequency is at or near zero and the program jumps to GZS (line 108) where R3 is loaded with a 1F (hex) to cause the CALC subroutine to load zero scale into R7/R6.

When (and if) TO is found to be low, the program jumps to line 95 and waits for that input to go high. Time out process is the same as above.

Now that the TO input is found high (if is is before the one second time out), the timer and carry bit are cleared in lines 97 through 100 (R3 is an extension of the timer).

At lines 101 through 107 we wait for one complete cycle at the TO input, with the timer/counter measuring that period, then return to line 80, where the CALC subroutine is called.

The CALC subroutine, starting at line 113, begins by initializing the word to send out (R7/R6) to zero, clearing the carry bit, checking to see if R3 indicates a time above one second, returning to line 81 if it does. Otherwise the program continues at line 26, where the program checks to see if the input frequency is beyond full scale (timer reading above 00 12 88 hex). If it is, R7/R6 is loaded with 12 88 hex (full scale of decimal 1,000). This value was chosen because it is sufficiently far from zero scale that it is easily discerned from zero scale on the display.

If the result is not to be full scale or zero scale, the program continues at line 140 with a shift and subtract divide routine. The dividend would be 1,000,000 (decimal) to convert back to frequency in Hertz (period measurements is in microseconds), but that would provide a maximum count of 200 at 200Hz, only one fifth of the full scale desired of 1,000. So we made the dividend to be 5,000,000 decimal, or 4C 4B 40 hex.

This algorithm is found in lines 156 through 192 and works as follows:

1. Clear a counter.
2. Rotate dividend until the first one is in the second MSB position. Since a code of 4C has already provides that, no shifting is necessary.
3. Rotate the divisor (the period in microseconds in this case) left until the first one is in the second MSB position, but the first byte is LESS THAN the first byte of the dividend. Increment the counter each time the divisor is rotated.
4. Initialize a counter to zero.
5. Rotate the quotient (answer) and dividend one bit left.
6. If first byte of quotient is smaller than the first byte of the quotient, jump to step 8.
7. Add one to the quotient and subtract the divisor from the dividend.
8. Decrement the counter and go to step 5 if it is not zero.

Once the CALC subroutine is completed, the program calls SENDIT from line 81 and jumps, ultimately, to the selected routine.

Routine 3
PROG3, beginning at line 194, measures the input period four times, then calculates the code to display that is the average of these four readings.

It starts by setting a counter for three readings, taking those three readings and storing them in memory, beginning at RAM address 20 hex, using register RO as an index register.

At line 212 the program takes a fourth reading, then adds the three previous readings to it in lines 213 through 227; and divides the sum by four (rotates right twice) in lines 229 through 239. The word to send out is then calculated from line 240 and sent to the ACMD, after which the program then looks for and jumps to the selected routine.

Routine 4
PROG4 begins at line 243 and displays the average of the current and last three words sent out.

RAM space used is first initialized to zero and a new reading is taken and a new word is calculated and saved. At lines 264 through 284, the new word is added to the last three readings and the average calculated and stored in RAM locations 28 and 29 (hex), and the average word is sent out.

At line 286, the program reads for the program selected and jumps to line 254 if this routine is selected, otherwise it goes to line 28.

ROUTINE 5

PROG5 begins at line 293 and, very simply, send in sequence the codes for 1/8 through full scale in 1/8 scale steps, with 500ms between steps. It then steps down to zero scale in 1/8 scale steps, then returns to line 28.

Routine 6
PROG6 begins at line 314 and does the same as PROG5, but steps in 1/4 scale increments.

Routine 7
PROG7 loads the code for 3/8 scale into R7/R6, sends it, waits 500ms, changes r& for 5/8 scale, sends it, waits for 500ms, then repeats this sequence 9 more times (for a total of ten times), waits 500ms, then returns the output to zero scale and the program jumps to line 28.

Controlling aire core meters with the 87C751 and SA 5775

AN426

```
                ;                                ACMD V3 DEMO                TT.20
                ;                             PROCESSOR:  87C751
                ;                                                            7-29-89
                ;
                ;   The purpose of this program is to drive version 3 of the ACMD (SA5775)
                ;   demonstration board.  The PROGRAM SELECT switch is used to select from
                ;   a choice of four routines.  Registers R7 and R6 contain the 10-bit word
                ;   that is send to the SA5775.
                ;
                $MOD751
0000                    ORG    0
                ;
0000 B02E               SJMP   START            ;RESET VECTOR
                ;
000B                    ORG    00BH             ;TIMER/COUNTER INTERRUPT ROUTINE
000B 0B                 INC    R3               ;INCREMENT R3 (3rd BYTE OF TIMER)
000C 740F               MOV    A,#0FH           ;TEST FOR TIME OUT (R3 > 0F)
000E 9B                 SUBB   A,R3             ;IF R3 > 0F, CARRY IS SET
000F 32                 RETI
                ;
0030                    ORG    30H              ;START OF PROGRAM
0030 7580FF     START:  MOV    P0,#0FFH         ;SET PORTS HIGH
0033 7590FF             MOV    P1,#0FFH
0036 75B0FF             MOV    P3,#0FFH
0039 7F00               MOV    R7,#0            ;CLEAR WORD TO SEND OUT
003B 7E00               MOV    R6,#0
003D 20B6FD     W:      JB     P3.6,W           ;WAIT FOR START BUTTON DEPRESS
0040 758BF0     READY:  MOV    RTL,#LOW(0-10000) ;SET TIMER REGISTER
0043 758DD8             MOV    RTH,#HIGH(0-10000);FOR 10ms TIME
0046 51D2               ACALL  RPS              ;READ PORT 3 FOR PROG SELECT
0048 90004C             MOV    DPTR,#JMPTBL     ;JMP ADDRESS TO DATA POINTER
004B 73                 JMP    @A+DPTR          ;GOTO APPROPRIATE ROUTINE
004C 015C       JMPTBL: AJMP   PROG0            ;RAMP UP AND BACK DOWN
004E 0168               AJMP   PROG1            ;STEP UP/DOWN W/ start PRESS
0050 017A               AJMP   PROG2            ;READ & DISPLAY SPEED
0052 2145               AJMP   PROG3            ;DISPLAY AVERAGE OF 4 NEW READINGS
0054 2186               AJMP   PROG4            ;DISPLAY AVERAGE OF LAST 4 READINGS
0056 21D3               AJMP   PROG5            ;ADVANCE TO FULL SCALE AND BACK IN 45 DEGREE STEPS
0058 21F3               AJMP   PROG6            ;ADVANCE TO FULL SCALE AND BACK IN 90 DEGREE STEPS
005A 4107               AJMP   PROG7            ;ALTERNATE DISPLAY BETWEEN 3/8 AND 5/8 SCALE TEN TIMES
005C            PROG0:
                ;   This routine increases word sent at the selected step size (INCREMENT SELECT)
                ;   and delay time (DELAY), up to full scale, waits 500ms, then decreases the
                ;   word sent at the selected step size and delay times until zero scale is reached.

005C 5128               ACALL  SO               ;SEND OUT INCREASING WORDS
005E 5121               ACALL  FULLSC           ;SET TO FULL SCALE
0060 51A5               ACALL  DLY500           ;WAIT 500ms
0062 5152               ACALL  SOD              ;SEND OUT DECREASING WORDS
0064 511B               ACALL  ZEROSC           ;RESET TO ZERO SCALE
0066 0130               AJMP   START            ;GO TO BEGINNING OF PROGRAM
006B            PROG1:
                ;
                ;        MANUAL INCREMENT/DECREMENT ROUTINE
                ;
                ;   This routine increases or decreases the sent out word, depending upon
                ;   the setting of the UP/DOWN switch, by an amount set by the INCREMENT
                ;   SELECT switch.  There is a wait of 200ms before again looking for
                ;   depression of the START/COUNT button to allow time to release this
                ;   button and switch bounce to settle.  The program then looks to see which
                ;   routine is selected and goes to that routine.
                ;
0068 30B50B             JNB    P3.5,DCX         ;GO AND COUNT DOWN IF SELECTED
006B 5130               ACALL  UP               ;INCREASE WORD
006D 51B5       DP1:    ACALL  SENDIT           ;SEND THE WORD
006F 519D               ACALL  DLY200           ;WAIT 200ms
0071 013D               AJMP   W                ;WAIT FOR COUNT BUTTON DEPRESS & SELECTED ROUTINE
0073 20B5F2     DCX:    JB     P3.5,PROG1       ;GO AND COUNT UP IF SELECTED
0076 515A               ACALL  DOWN             ;DECREASE WORD
```

Controlling aire core meters with the 87C751 and SA 5775

AN426

```
0078 80F3     70          SJMP   DP1
007A          71   PROG2:
              72   ;
              73   ;        READ TIME INPUT AND DISPLAY "SPEED"
              74   ;
              75   ;   This routine measures the period of the square wave at the T0 input and
              76   ;   sends out a word that is inversely proportional to 5 times that period,
              77   ;   providing a display proportional to frequency.
              78   ;
007A 1182     79          ACALL  MEAS              ;MEASURE THE INPUT PERIOD
007C 11C5     80          ACALL  CALC              ;CALCULATE THE WORD TO SEND
007E 51B5     81          ACALL  SENDIT            ;SEND OUT THE WORD
0080 0140     82          AJMP   READY
0082 C28C     83   MEAS:  CLR    TR                ;HALT TIMER
0084 C28D     84          CLR    TF                ;CLEAR TIMER FLAG
0086 758B00   85          MOV    RTL,#0            ;SET TIMER REGISTERS
0089 758D00   86          MOV    RTH,#0
008C 758A00   87          MOV    TL,#0             ;SET TIMER
008F 758C00   88          MOV    TH,#0
0092 7B00     89          MOV    R3,#0             ;CLEAR TIMER 3RD BYTE
0094 C3       90          CLR    C
0095 D28C     91          SETB   TR                ;START TIMER
0097 75A882   92          MOV    IE,#82H           ;ENABLE TIMER INTERRUPT
009A 4021     93   W20:   JC     GZS               ;JUMP IF R3 > 0F
009C 2097FB   94          JB     P1.7,W20          ;WAIT FOR T0 INPUT LOW
009F 401C     95   W21:   JC     GZS               ;JUMP IF R3 > 0F
00A1 3097FB   96          JNB    P1.7,W21          ;WAIT FOR T0 INPUT HIGH
00A4 758A00   97          MOV    TL,#0             ;RESET TIMER
00A7 758C00   98          MOV    TH,#0
00AA 7B00     99          MOV    R3,#0
00AC C3      100          CLR    C                 ;CLEAR CARRY/BORROW
00AD 4008    101   W22:   JC     HT                ;JUMP IF TIME UP (CARRY SET)
00AF 2097FB  102          JB     P1.7,W22          ;WAIT FOR T0 LOW
00B2 4003    103   W23:   JC     HT                ;JUMP IF TIME UP (CARRY SET)
00B4 3097FB  104          JNB    P1.7,W23          ;WAIT FOR T0 HIGH AGAIN
00B7 C28C    105   HT:    CLR    TR                ;HALT TIMER
00B9 75A800  106          MOV    IE,#0             ;DISABLE ALL INTERRUPTS
00BC 22      107          RET
00BD 7B1F    108   GZS:   MOV    R3,#1FH           ;SET FOR ZERO SCALE
00BF 22      109          RET
00C0 7F03    110   GFS:   MOV    R7,#03
00C2 7EE8    111          MOV    R6,#0E8H
00C4 22      112          RET
00C5         113   CALC:
             114   ;
             115   ;   This subroutine calculates the 10-bit word to send as a function fo what
             116   ;   is in R3, TH & TL.  The 10-bit word is developed and left in registers
             117   ;   R7 and R6 for use by SENDIT subroutine.
             118   ;
00C5 7F00    119          MOV    R7,#0             ;INITIALIZE QUOTIENT
00C7 7E00    120          MOV    R6,#0
00C9 C3      121          CLR    C                 ;CLEAR CARRY/BORROW
00CA 740F    122          MOV    A,#0FH            ;CHECK FOR ZERO SCALE
00CC 9B      123          SUBB   A,R3
00CD 5001    124          JNC    NZS               ;JUMP IF NOT ZERO SCALE
00CF 22      125          RET
00D0 E58A    126   NZS:   MOV    A,TL              ;CHECK FOR FULL SCALE
00D2 9488    127          SUBB   A,#88H
00D4 E58C    128          MOV    A,TH
00D6 9413    129          SUBB   A,#13H
00D8 EB      130          MOV    A,R3
00D9 9400    131          SUBB   A,#0
00DB 40E3    132          JC     GFS
00DD 752E4C  133          MOV    2EH,#4CH          ;SET DIVIDEND TO 5,000,000
00E0 752F4B  134          MOV    2FH,#4BH
00E3 753040  135          MOV    30H,#40H
00E6 7C00    136          MOV    R4,#0             ;CLEAR DIVIDE COUNTER
00E8 8B2B    137          MOV    2BH,R3            ;MOVE READING TO MEMORY (DIVISOR)
00EA 858C2C  138          MOV    2CH,TH
```

Controlling aire core meters with the 87C751 and SA 5775

AN426

```
00ED 858A2D  139          MOV    2DH,TL
00F0 C3      140  ROTL:    CLR    C              ;BRING DIVISOR BE JUST LESS THAN DIVIDEND
00F1 E52E    141          MOV    A,2EH
00F3 952B    142          SUBB   A,2BH
00F5 4014    143          JC     DIV24          ;JUMP IF SHIFTING WOULD MAKE DIVISOR > DIVIDEND
00F7 6012    144          JZ     DIV24          ;JUMP IF DIVISOR & DIVIDEND MS BYTES EQUAL BEFORE SHIFT
00F9 E52D    145          MOV    A,2DH          ;SHIFT DIVISOR TO LEFT
00FB 33      146          RLC    A
00FC F52D    147          MOV    2DH,A
00FE E52C    148          MOV    A,2CH
0100 33      149          RLC    A
0101 F52C    150          MOV    2CH,A
0103 E52B    151          MOV    A,2BH
0105 33      152          RLC    A
0106 F52B    153          MOV    2BH,A
0108 0C      154          INC    R4
0109 80E5    155          SJMP   ROTL
010B C3      156  DIV24:   CLR    C
010C EE      157          MOV    A,R6           ;ROTATE QUOTIENT LEFT
010D 33      158          RLC    A
010E FE      159          MOV    R6,A
010F EF      160          MOV    A,R7
0110 33      161          RLC    A
0111 FF      162          MOV    R7,A
0112 C3      163          CLR    C              ;ROTATE DIVIDEND LEFT
0113 E530    164          MOV    A,30H
0115 33      165          RLC    A
0116 F530    166          MOV    30H,A
0118 E52F    167          MOV    A,2FH
011A 33      168          RLC    A
011B F52F    169          MOV    2FH,A
011D E52E    170          MOV    A,2EH
011F 33      171          RLC    A
0120 F52E    172          MOV    2EH,A
0122 C3      173          CLR    C              ;TEST SUBTRACT MOST SIGNIFICANT BYTES
0123 952B    174          SUBB   A,2BH
0125 401B    175          JC     ZERO           ;JUMP IF QUOTIENT MS BYTE < DIVISOR MS BYTE
0127 7401    176          MOV    A,#1           ;ADD 1 TO QUOTIENT
0129 2E      177          ADD    A,R6
012A FE      178          MOV    R6,A
012B EF      179          MOV    A,R7
012C 3400    180          ADDC   A,#0
012E FF      181          MOV    R7,A
012F C3      182          CLR    C              ;SUBTRACT DIVISOR FROM DIVIDEND
0130 E530    183          MOV    A,30H
0132 952D    184          SUBB   A,2DH
0134 F530    185          MOV    30H,A
0136 E52F    186          MOV    A,2FH
0138 952C    187          SUBB   A,2CH
013A F52F    188          MOV    2FH,A
013C E52E    189          MOV    A,2EH
013E 952B    190          SUBB   A,2BH
0140 F52E    191          MOV    2EH,A
0142 DCC7    192  ZERO:    DJNZ   R4,DIV24
0144 22      193          RET
0145         194  PROG3:
             195  ;
             196  ;              DISPLAY AVERAGE OF FOUR NEW READINGS
             197  ;
             198  ;   This routine reads the period of the T0 input four times, then displays the
             199  ;   "speed" corresponding to the average of these four readings.
             200  ;
0145 7903    201          MOV    R1,#3          ;SET FOR 3 READINGS
0147 7820    202          MOV    R0,#20H        ;SET INDEX REGISTER FOR BOTTOM
0149 1182    203  P30:    ACALL  MEAS           ;TAKE 3 READINGS AND SAVE THEM
014B EB      204          MOV    A,R3
014C F6      205          MOV    @R0,A
014D 08      206          INC    R0
014E A68C    207          MOV    @R0,TH
```

Controlling aire core meters with the 87C751 and SA 5775

AN426

```
0150 08       208          INC    R0
0151 A68A     209          MOV    @R0,TL
0153 08       210          INC    R0
0154 D9F3     211          DJNZ   R1,P30
0156 1182     212          ACALL  MEAS              ;TAKE A 4TH READING, LEAVING IN R3,TH,TL
0158 7828     213          MOV    R0,#28H           ;SET INDEX REGISTER FOR TOP
015A 7903     214          MOV    R1,#3             ;SET COUNTER TO ADD FIRST 3 READINGS TO LAST ONE
015C E58A     215  P31:    MOV    A,TL              ;ADD FIRST THREE READINGS TO THE LAST ONE
015E 26       216          ADD    A,@R0
015F F58A     217          MOV    TL,A
0161 18       218          DEC    R0
0162 E58C     219          MOV    A,TH
0164 36       220          ADDC   A,@R0
0165 F58C     221          MOV    TH,A
0167 18       222          DEC    R0
0168 EB       223          MOV    A,R3
0169 36       224          ADDC   A,@R0
016A FB       225          MOV    R3,A
016B 18       226          DEC    R0
016C D9EE     227          DJNZ   R1,P31
016E 7902     228          MOV    R1,#2
0170 EB       229  P32:    MOV    A,R3              ;DIVIDE BY 4 (ROTATE RIGHT TWICE) FOR AVERAGE
0171 C3       230          CLR    C
0172 13       231          RRC    A
0173 FB       232          MOV    R3,A
0174 E58C     233          MOV    A,TH
0176 13       234          RRC    A
0177 F58C     235          MOV    TH,A
0179 E58A     236          MOV    A,TL
017B 13       237          RRC    A
017C F58A     238          MOV    TL,A
017E D9F0     239          DJNZ   R1,P32
0180 11C5     240          ACALL  CALC              ;CALCULATE THE WORD
0182 51B5     241          ACALL  SENDIT            ;SEND OUT THE WORD
0184 0140     242          AJMP   READY             ;GO TO SELECTED ROUTINE
0186          243  PROG4:
              244  ;
              245  ;        DISPLAY AVERAGE OF LAST FOUR WORDS SENT OUT
              246  ;
              247  ;   This routine sends out the average of the last four readings sent out.
              248  ;
0186 7827     249          MOV    R0,#27H
0188 7600     250  P4:     MOV    @R0,#0
018A 18       251          DEC    R0
018B B81FFA   252          CJNE   R0,#1FH,P4
018E 7820     253  P4A:    MOV    R0,#20H
0190 1182     254  P40:    ACALL  MEAS              ;MEASURE PERIOD
0192 11C5     255          ACALL  CALC              ;CALCULATE THE CODE
0194 EF       256          MOV    A,R7              ;SAVE THE CODE
0195 F6       257          MOV    @R0,A
0196 08       258          INC    R0
0197 EE       259          MOV    A,R6
0198 F6       260          MOV    @R0,A
0199 752800   261          MOV    28H,#0            ;INITIALIZE THE WORD TO SEND
019C 752900   262          MOV    29H,#0
019F 7927     263          MOV    R1,#27H
01A1 E529     264  P41:    MOV    A,29H             ;ADD TOGETHER LAST 4 RESULTS
01A3 C3       265          CLR    C
01A4 27       266          ADD    A,@R1
01A5 F529     267          MOV    29H,A
01A7 E528     268          MOV    A,28H
01A9 19       269          DEC    R1
01AA 37       270          ADDC   A,@R1
01AB F528     271          MOV    28H,A
01AD 19       272          DEC    R1
01AE B91FF0   273          CJNE   R1,#1FH,P41
01B1 7902     274          MOV    R1,#2
01B3 C3       275  P42:    CLR    C
01B4 E528     276          MOV    A,28H
```

Controlling aire core meters with the 87C751 and SA 5775

AN426

```
01B6 13      277          RRC    A
01B7 F528    278          MOV    28H,A
01B9 E529    279          MOV    A,29H
01BB 13      280          RRC    A
01BC F529    281          MOV    29H,A
01BE D9F3    282          DJNZ   R1,P42
01C0 AF28    283          MOV    R7,28H
01C2 AE29    284          MOV    R6,29H
01C4 51B5    285          ACALL  SENDIT          ;SEND OUT THE WORD
01C6 51D2    286          ACALL  RPS             ;READ PROGRAM SELECT
01C8 B40806  287          CJNE   A,#8,N4         ;JUMP TO N4 (& "READY") IF PROGRAM 4 NOT SELECTED
01CB 08      288          INC    R0
01CC B828C1  289          CJNE   R0,#28H,P40     ;GOTO P40 IF R0 NOT 28 (HEX)
01CF 80BD    290          SJMP   P4A
01D1 0140    291  N4:     AJMP   READY
             292  ;
             293  PROG5:
             294  ;
             295  ;   This routine advances the display in 45 degree steps to full scale, then steps down
             296  ;   to zero in 45 degree steps.  There is a 500ms delay between steps.
             297  ;
01D3 7F00    298          MOV    R7,#0
01D5 7E7F    299  P5:     MOV    R6,#07FH
01D7 51B1    300          ACALL  SD500           ;SEND THE WORD AND WAIT 500ms
01D9 7EFF    301          MOV    R6,#0FFH
01DB 51B1    302          ACALL  SD500           ;SEND THE WORD AND WAIT 500ms
01DD 0F      303          INC    R7
01DE BF04F4  304          CJNE   R7,#4,P5
01E1 7F03    305          MOV    R7,#3
01E3 7EFF    306  LP5:    MOV    R6,#0FFH
01E5 51B1    307          ACALL  SD500           ;SEND THE WORD AND WAIT 500ms
01E7 7E7F    308          MOV    R6,#7FH
01E9 51B1    309          ACALL  SD500
01EB 1F      310          DEC    R7
01EC BFFFF4  311          CJNE   R7,#0FFH,LP5
01EF 511B    312          ACALL  ZEROSC          ;RETURN TO ZERO
01F1 013D    313          AJMP   W               ;WAIT FOR KEY PRESS
01F3         314  PROG6:
             315  ;
             316  ;   This routine advances the display in 90 degree steps to full scale, then steps down
             317  ;   to zero in 90 degree steps.  There is a 500ms delay between steps.
             318  ;
01F3 7EFF    319          MOV    R6,#0FFH
01F5 7F00    320          MOV    R7,#0
01F7 51B1    321  LP6:    ACALL  SD500           ;SEND THE WORD AND WAIT 500ms
01F9 0F      322          INC    R7
01FA BF04FA  323          CJNE   R7,#4,LP6
01FD 1F      324  LP6A:   DEC    R7
01FE 51B1    325          ACALL  SD500           ;SEND THE WORD AND WAIT 500ms
0200 BF00FA  326          CJNE   R7,#0,LP6A
0203 511B    327          ACALL  ZEROSC          ;RETURN TO ZERO
0205 013D    328          AJMP   W               ;WAIT FOR KEY PRESS
0207         329  PROG7:
             330  ;
             331  :   This routine alternates between 3/8 and 5/8 scale ten times with 300ms delay
             332  ;   between steps, then waits 500ms before returning display to zero scale.
             333  ;
0207 7A0A    334          MOV    R2,#10          ;SET COUNTER
0209 7E7F    335  PR7:    MOV    R6,#07FH
020B 7F01    336          MOV    R7,#1
020D 51AD    337          ACALL  SD300           ;SEND OUT THE WORD AND WAIT 300ms
020F 7F02    338          MOV    R7,#2
0211 51AD    339          ACALL  SD300           ;SEND OUT THE WORD AND WAIT 300ms
0213 DAF4    340          DJNZ   R2,PR7          ;DO IT 10 TIMES
0215 51A5    341          ACALL  DLY500          ;WAIT 500ms
0217 511B    342          ACALL  ZEROSC          ;RESET TO ZERO SCALE
0219 0130    343          AJMP   START           ;LOOK FOR VALID PROGRAM
             344  ;
             345  ;
```

Controlling aire core meters with the 87C751 and SA 5775


```
                346  ;               SUBROUTINES
                347  ;
                348  ;
021B 7F00       349  ZEROSC: MOV     R7,#0              ;RESET METER TO ZERO SCALE
021D 7E00       350         MOV     R6,#0
021F 4125       351         AJMP    RST
0221 7F03       352  FULLSC: MOV     R7,#03H            ;SET METER TO FULL SCALE
0223 7EFF       353         MOV     R6,#0FFH
0225 51B5       354  RST:   ACALL   SENDIT
0227 22         355
0228            356  SO:
                357  ;
                358  ;   This subroutine sends increasing 10-bit words in registers R7 & R6 to the ACMD.
                359  ;
0228 51B5       360         ACALL   SENDIT             ;WRITE THE 10-BIT WORD TO ACMD
022A 5130       361         ACALL   UP                 ;INCREASE THE WORD VALUE
022C 30E2F9     362         JNB     ACC.2,SO           ;JUMP IF BIT 2 NOT SET
022F 22         363         RET
0230            364  UP:
                365  ;
                366  ;   This subroutine waits for a period of time = 10ms X DELAY read un, then
                367  ;   increases the 10-bit word by the INCREMENT SELECT amount.
                368  ;
0230 E590       369         MOV     A,P1               ;READ DELEY
0232 F4         370         CPL     A                  ;COMPLEMENT ACC
0233 540F       371         ANL     A,#0FH             ;MASK OFF UPPER 4 BITS
0235 23         372         RL      A
0236 23         373         RL      A
0237 F9         374         MOV     R1,A
0238 B90002     375         CJNE    R1,#0,D10          ;JUMP IF DELAY SET FOR ZERO
023B 8006       376         SJMP    NODLY
023D 7B01       377  D10:    MOV     R3,#1              ;SET FOR 1 X 10ms DELAY
023F 5195       378  D10A:   ACALL   DLY10MS           ;DELAY 10MS x DELAY
0241 D9FC       379         DJNZ    R1,D10A
0243 E5B0       380  NODLY:  MOV     A,P3               ;READ INCREMENT SELECT
0245 F4         381         CPL     A                  ;COMPLEMENT ACC
0246 5403       382         ANL     A,#3               ;MASK OFF UPPER 6 BITS
0248 23         383         RL      A
0249 23         384         RL      A
024A 23         385         RL      A
024B 04         386         INC     A
024C 2E         387         ADD     A,R6               ;ADD INCREMENT TO R6
024D FE         388         MOV     R6,A               ;SAVE IT
024E E4         389         CLR     A
024F 3F         390         ADDC    A,R7               ;ADD CARRY TO R7
0250 FF         391         MOV     R7,A               ;SAVE IT
0251 22         392         RET
0252            393  SOD:
                394  ;
                395  ;   This subroutine sends out decreasing words at the rate set by DELAY and
                396  ;   step size determined by INCREMENT SELECT.
                397  ;
0252 51B5       398         ACALL   SENDIT             ;SEND OUT THE PRESENT WORD
0254 515A       399         ACALL   DOWN               ;DECREASE THE WORD
0256 50FA       400         JNC     SOD                ;DO IT AGAIN IF CARRY NOT SET
0258 411B       401         AJMP    ZEROSC
025A            402  DOWN:
                403  ;
                404  ;   Waits for 10ms x DELAY pot setting, then sends out decreasing values of words
                405  ;   in step sizes of 8 x INCREMENT SELECT + 1.
                406  ;
025A E590       407         MOV     A,P1               ;READ DELAY
025C F4         408         CPL     A                  ;COMPLEMENT ACC
025D 540F       409         ANL     A,#0FH             ;MASK OFF UPPER FOUR BITS
025F 23         410         RL      A
0260 23         411         RL      A
0261 F9         412         MOV     R1,A               ;SAVE DELAY
0262 B90002     413         CJNE    R1,#0,D10S         ;JUMP IF DELAY SET FOR ZERO
0265 8004       414         SJMP    NDD
```

Controlling aire core meters with the 87C751 and SA 5775

AN426

```
0267 5195    415  D10S:   ACALL  DLY10MS          ;DELAY 10ms x (DELAY +1)
0269 D9FC    416          DJNZ   R1,D10S
026B E5B0    417  NDD:    MOV    A,P3              ;READ INCREMENT SELECT
026D F4      418          CPL    A                 ;COMPLEMENT ACC
026E 5403    419          ANL    A,#3              ;MASK OFF UPPER 6 BITS
0270 23      420          RL     A                 ;MULTIPLY BY 8
0271 23      421          RL     A
0272 23      422          RL     A
0273 04      423          INC    A                 ;INSURE MINIMUM STEP
0274 C3      424          CLR    C                 ;CLEAR CARRY FOR SUBTRACTION
0275 CE      425          XCH    A,R6
0276 9E      426          SUBB   A,R6              ;SUBTRACT INCREMENT FROM R6
0277 CE      427          XCH    A,R6              ;SAVE IT
0278 E4      428          CLR    A                 ;CLEAR ACCUM FOR SUBTRACTION
0279 CF      429          XCH    A,R7
027A 9F      430          SUBB   A,R7              ;SUBTRACT BORROW FROM R7
027B 5403    431          ANL    A,#3              ;INSURE MAXIMUM WORD
027D CF      432          XCH    A,R7              ;SAVE IT
027E 22      433          RET
027F 00      434  DELAY:  NOP                      ;3µs DELAY
0280 22      435          RET
0281         436  DMS10:
             437  ;
             438  ;   Produces a delay of 10ms x the value in R3.
             439  ;   Destroys R3 and timer readings.
             440  ;
             441  ;
0281 758AF0  442          MOV    TL,#LOW,(0-10000) ;LOAD TIMER FOR 10ms DELAY
0284 758CD8  443          MOV    TH,#HIGH(0-10000)
0287 C28D    444          CLR    TF                ;CLEAR TIMER FLAG
0289 D28C    445          SETB   TR                ;START TIMER
028B 308DFD  446  MS10W:  JNB    TF,MS10W          ;WAIT FOR TIMER FLAG TO BE SET
028E C28D    447          CLR    TF                ;CLEAR TIMER FLAG
0290 DBF9    448          DJNZ   R3,MS10W          ;WAIT RS x 10ms
0292 C28C    449          CLR    TR                ;STOP TIMER
0294 22      450          RET
             451  ;
0295 7B01    452  DLY10MS: MOV   R3,#1             ;SET R3 FOR 10ms WAIT
0297 80EB    453          SJMP   DMS10             ;WAIT 10ms
             454  ;
0299 7B0A    455  DLY100:  MOV   R3,#10            ;SET R3 FOR 100ms WAIT
029B 80E4    456          SJMP   DMS10             ;WAIT 100ms
             457  ;
029D 7B14    458  DLY200:  MOV   R3,#20            ;SET R3 FOR 200ms WAIT
029F 80E0    459          SJMP   DMS10             ;WAIT 200ms
             460  ;
02A1 7B1E    461  DLY300:  MOV   R3,#30            ;SET R3 FOR 300ms WAIT
02A3 80DC    462          SJMP   DMS10             ;WAIT 300ms
             463  ;
02A5 7B32    464  DLY500:  MOV   R3,#50            ;SET R3 FOR 500ms WAIT
02A7 80D8    465          SJMP   DMS10             ;WAIT 500ms
             466  ;
02A9 51B5    467  SD200:   ACALL  SENDIT          ;SEND THE WORD
02AB 80F0    468          SJMP   DLY200           ;WAIT 200ms
             469  ;
02AD 51B5    470  SD300:   ACALL  SENDIT          ;SEND THE WORD
02AF 80F0    471          SJMP   DLY300           ;WAIT 200ms
             472  ;
02B1 51B5    473  SD500:   ACALL  SENDIT          ;SEND THE WORD
02B3 80F0    474          SJMP   DLY500           ;WAIT 500ms
             475  ;
02B5         476  SENDIT:
             477  ;
             478  ;   This subroutine sends out a single word locate4d in R7 and R6.
             479  ;   Accumulator, R0 and R1 are destroyed.
             480  ;
02B5 D282    481          SETB   P0.2             ;SET CS HIGH
02B7 7902    482          MOV    R1,#02           ;SET COUNTER FOR 2 BITS OF R7
02B9 EF      483          MOV    A,R7             ;MOVE R7 TO A FOR SEND OUT
```

Controlling aire core meters with the 87C751 and SA 5775


```
02BA 13     484          RRC    A                ;ALIGN R7 FOR SEND OUT
02BB 13     485          RRC    A
02BC 13     486          RRC    A
02BD 51C7   487          ACALL  SEND1            ;SEND OUT UPPER TWO BITS
02BF 7908   488          MOV    R1,#8            ;SET COUNTER FOR R6 SEND OUT
02C1 EE     489          MOV    A,R6             ;MOVE R6 TO ACCUM
02C2 51C7   490          ACALL  SEND1            ;SEND OUT LOWER 8 BITS
02C4 C282   491          CLR    P0.2             ;LOAD ACMD
02C6 22     492          RET
02C7        493  SEND1:
            494  ;
            495  ;   This subroutine sends [R1] number of bits of the accumulator, starting
            496  ;   with the MSB over the IIC port.
            497  ;   Accumulator, R0 and R1 are destroyed.
            498  ;
02C7 33     499          RLC    A                ;ROTATE BIT TO CARRY
02C8 9281   500          MOV    P0.1,C           ;MOVE CARRY TO DATA OUT
02CA C280   501          CLR    P0.0             ;CLOCK LOW
02CC 00     502          NOP
02CD D280   503          SETB   P0.0             ;CLOCK HIGH
02CF D9F6   504          DJNZ   R1,SEND1         ;SEND NEXT BIT TILL DONE
02D1 22     505          RET
            506  ;
02D2 E5B0   507  RPS:    MOV    A,P3             ;READ PORT 3 FOR PROGRAM SELECT
02D4 F4     508          CPL    A                ;COMPLEMENT ACC
02D5 03     509          RR     A                ;ROTATE TO LSB's & MULT BY 2
02D6 540E   510          ANL    A,#0EH           ;MASK FOR PROGRAM SELECT * 2
02D8 DD     511          RET
            512  END

ASSEMBLY COMPLETE, 0 ERRORS FOUND
```

```
ACC ..... D ADDR   00E0H     PREDEFINED
CALC ..... C ADDR   00C5H
D10 ..... C ADDR   023DH
D10A ..... C ADDR   023FH
D10S ..... C ADDR   0267H
DCX ..... C ADDR   0073H
DELAY ..... C ADDR   027FH     NOT USED
DIV24 ..... C ADDR   010BH
DLY100 ..... C ADDR   0299H     NOT USED
DLY10MS ..... C ADDR   0295H
DLY200 ..... C ADDR   029DH
DLY300 ..... C ADDR   02A1H
DLY500 ..... C ADDR   02A5H
DMS10 ..... C ADDR   0281H
DOWN ..... C ADDR   025AH
DP1 ..... C ADDR   006DH
FULLSC ..... C ADDR   0221H
GFS ..... C ADDR   00C0H
GZS ..... C ADDR   00BDH
HT ..... C ADDR   00B7H
IE ..... D ADDR   00A8H     PREDEFINED
JMPTBL ..... C ADDR   004CH
LP5 ..... C ADDR   01E3H
LP6 ..... C ADDR   01F7H
LP6A ..... C ADDR   01FDH
MEAS ..... C ADDR   0082H
MS10W ..... C ADDR   028BH
N4 ..... C ADDR   01D1H
NDD ..... C ADDR   026BH
NODLY ..... C ADDR   0243H
NZS ..... C ADDR   00D0H
P0 ..... D ADDR   0080H     PREDEFINED
P1 ..... D ADDR   0090H     PREDEFINED
P3 ..... D ADDR   00B0H     PREDEFINED
P30 ..... C ADDR   0149H
P31 ..... C ADDR   015CH
P32 ..... C ADDR   0170H
P4 ..... C ADDR   0188H
P40 ..... C ADDR   0190H
P41 ..... C ADDR   01A1H
P42 ..... C ADDR   01B3H
P4A ..... C ADDR   018EH
P5 ..... C ADDR   01D5H
PR7 ..... C ADDR   0209H
PROG0 ..... C ADDR   005CH
PROG1 ..... C ADDR   0068H
PROG2 ..... C ADDR   007AH
PROG3 ..... C ADDR   0145H
PROG4 ..... C ADDR   0186H
PROG5 ..... C ADDR   01D3H
PROG6 ..... C ADDR   01F3H
PROG7 ..... C ADDR   0207H
READY ..... C ADDR   0040H
ROTL ..... C ADDR   00F0H
RPS ..... C ADDR   02D2H
RST ..... C ADDR   0225H
RTH ..... D ADDR   008DH     PREDEFINED
RTL ..... D ADDR   008BH     PREDEFINED
SD200 ..... C ADDR   02A9H     NOT USED
SD300 ..... C ADDR   02ADH
SD500 ..... C ADDR   02B1H
SEND1 ..... C ADDR   02C7H
SENDIT ..... C ADDR   02B5H
SO ..... C ADDR   0228H
SOD ..... C ADDR   0252H
START ..... C ADDR   0030H
```

Controlling aire core meters with the 87C751 and SA 5775


```
TF ............ B ADDR   008DH     PREDEFINED
TH ............ D ADDR   008CH     PREDEFINED
TL ............ D ADDR   008AH     PREDEFINED
TR ............ B ADDR   008CH     PREDEFINED
UP ............ C ADDR   0230H
W ............. C ADDR   003DH
W20 ........... C ADDR   009AH
W21 ........... C ADDR   009FH
W22 ........... C ADDR   00ADH
W23 ........... C ADDR   00B2H
ZERO .......... C ADDR   0142H
ZEROSC ........ C ADDR   021BH
```

Timer I in non-I²C applications of the 83/87C751/752 microcontrollers

AN427

The small package of the 83/87C751 and 83/87C752 microcontrollers includes two hardware-implemented timers: a 16-bit programmable timer, and a 10-bit fixed-rate timer. The programmable timer is available for the application program, and its operation is similar to the timer/counter of the 80C51 timer in mode 2. The fixed-rate timer, Timer I, is typically employed as a watchdog timer for the I²C port communications and is not available for other uses.

In applications which do not take advantage of the I²C communications capability, the "silicon real estate" taken by Timer I is not necessarily lost—it can be used as a fixed-rate timer by the application. This timer can become useful in various cases, such as simple control applications that need a delay while doing some software activities in parallel, or generating a free-running repetitive waveform where the exact timing is not important. Another type of application is a watchdog timer prompting the user about unexpected operation of a system or its hardware, or resetting a program that "lost track."

TIMER I IMPLEMENTATION

Timer I is clocked once per machine cycle, which is the oscillator frequency divided by 12. The timer operation is enabled by setting the TIRUN bit (bit 4) in the I2CFG register. Writing a 0 into the TIRUN bit will stop and clear the timer. The timer is 10 bits wide, and when it reaches the terminal count of 1024 it carries out and sets the Timer I interrupt flag. An interrupt will occur if the Timer I interrupt is enabled by bit ETI (bit 4) of the Interrupt Enable (IE) register, and global interrupts are enabled by bit EA (bit 7) of the same IE register.

The vector address for the Timer I interrupt is 1B hex, and the interrupt service routine must start at this address. As with all 8051 family microcontrollers, only the Program Counter is pushed onto the stack upon interrupt (other registers that are used both by the interrupt service routine and elsewhere must be explicitly saved). The Timer I interrupt flag is cleared by setting the CLRTI bit (bit 5) of the I2CFG register.

Note that when the I²C interface is not operating—SLAVEN, MASTRQ, and MASTER bits are all 0—the I²C hardware does not affect Timer I. The SCL and SDA pins can be used as I/O pins, and the activity of these pins will not cause the timer to run, stop, or reset. Upon hardware reset of the microcontroller, the SLAVEN, MASTRQ, and MASTER bits are all reset, so the programmer does not have to worry about interaction between the SDA/SCL pins and the timer.

FIXED-RATE TIMER

The first programming example demonstrates simple fixed-rate operation. Upon reset, interrupts are enabled, and Timer I is started. A wait loop simulates the "application" program. The demonstration service routine simply sets a flag—in real life it could do something more useful, such as toggling an output pin. Note that the interrupt flag is cleared by setting CLRTI prior to returning from the service routine. Upon overflow, the timer will go on running, as the TIRUN bit is still set, so the interrupts will be spaced exactly 1024 clock cycles apart. If the service routine would toggle an output pin instead of setting a flag, its output would be a square wave with a period of 2048 cycles. For an application that demands a "one-shot" delay only, the service routine should clear the TIRUN bit in order to avoid subsequent interrupts.

WATCHDOG TIMER

A watchdog timer mechanism is typically applied in order to detect "abnormal" behavior of hardware. If the microcontroller operates in a very noisy environment, there might be a fear of the program "running wild" as a result of extremely violent EMI interference. In such a case, a watchdog may take care to reset the microcontroller when the Timer I interrupt occurs. This could be applied in application programs with a repetitive nature—the software needs to reset the timer within 1024 machine cycles of the last reset.

In a system where something is supposed to occur regularly—for example, an interrupt for an external event—the watchdog is designed to "bite" when the hardware "sleeps" and the expected "something" does not happen for too long a time. The timer is allowed to run continuously, but when the expected event occurs, it resets the timer back to 0. When the timer is reset within 1024 cycles of the last reset, the application runs normally. If the event does not occur, the Timer I interrupt service routine will be activated to take care of the exception.

The second programming example demonstrates the watchdog. Upon Reset, the TIRUN bit, ETI, and global interrupts are enabled. The watchdog timer is reset and restarted by the small subroutine WdRst. The application is simulated by a loop of delays. Delay 1 is less than 1024 cycles, and when WdRst is called within Delay 1 intervals, no Timer I interrupt occurs. This represents normal operation of a "real life" application. When the delay from last reset is greater than 1024 cycles—representing a hardware exception—the interrupt will occur. The service routine for the watchdog is somewhat unusual, as it does not return to the program location where the interrupt occurred. Instead, the operation of the microcontroller is restarted at Reset. Upon entering the service routine, the interrupt is cleared and the timer is reset. Because execution does not return to the interrupted program with a RETI instruction, the interrupt pending flag is cleared by a call to a dummy subroutine XRETI. The program is restarted at Reset with a regular AJMP instruction. The stack pointer is explicitly reinitialized for the warm reset, so there is no danger of stack overflow upon repeated watchdog invocations.

Timer I in non-I²C applications of the 83/87C751/752 microcontrollers

AN427

```
TIINT                    Timer I Fixed Rate Timer                    11/06/90   PAGE 1

                 ;
                  ***************************************************************************

                 ;                    Timer I Fixed Rate Timer Usage

                 ;This program demonstrates how to activate Timer I on the 83C751 or
                 ;83C752 microcontrollers as a fixed rate timer when the I2C port is not
                 ;used. Once activated, Timer I will generate an interrupt every 1024
                 ;machine cycles. The I2C bus pins SCL and SDA may be used as open drain
                 ;outputs.

                 ;
                  ***************************************************************************

                 $MOD7 751
                 $Title(Timer I Fixed Rate Timer)
                 $Date(11/06/90)
                 $Debug


0020             Flags     DATA  20h            ;Flag byte
0000             TstFlag   BIT   Flags.0        ;Timer I flag.


0000                       ORG   0
0000 0120                  AJMP  Reset

001B                       ORG   1Bh            ;Timer I interrupt.
001B D200        TimerI:   SETB  TstFlag        ;Set flag to indicate a Timer I interrupt.
001D D2DD                  SETB  CLRTI          ;Clear Timer I to allow it to restart.
001F 32                    RETI


0020 D2AB        Reset:    SETB  ETI            ;Enable Timer I interrupt.
0022 D2AF                  SETB  EA             ;Enable global interrupts.
0024 D2DC                  SETB  TIRUN          ;Start Timer I.

0026 C200        Loop:     CLR   TstFlag        ;Initialize interrupt flag.
0028 3000FD      Wait:     JNB   TstFlag,Wait   ;Wait for Timer I interrupt.
002B 80F9                  SJMP  Loop

                           END

ASSEMBLY COMPLETE, 0 ERRORS FOUND
```

Timer I in non-I²C applications of the 83/87C751/752 microcontrollers

AN427

```
TIINT                          Timer I Fixed Rate Timer                    11/06/90   PAGE 2

CLRTI. . . . . . . . . . . . . B ADDR  00DDH  PREDEFINED
EA . . . . . . . . . . . . . . B ADDR  00AFH  PREDEFINED
ETI. . . . . . . . . . . . . . B ADDR  00ABH  PREDEFINED
FLAGS. . . . . . . . . . . . . D ADDR  0020H
LOOP . . . . . . . . . . . . . C ADDR  0026H
RESET. . . . . . . . . . . . . C ADDR  0020H
TIMERI . . . . . . . . . . . . C ADDR  001BH  NOT USED
TIRUN. . . . . . . . . . . . . B ADDR  00DCH  PREDEFINED
TSTFLAG. . . . . . . . . . . . B ADDR  0000H
WAIT . . . . . . . . . . . . . C ADDR  0028H
```

Timer I in non-I²C applications of the 83/87C751/752 microcontrollers

AN427

```
TIWD                                Timer I Watchdog                                11-06-90   PAGE 1

                ;*******************************************************************************

                ;                    Timer I Watchdog Timer Usage

                ;This program demonstrates how to use Timer I on the 83C751 or 83C752
                ;microcontrollers as a watchdog timer when the I2C port is not used.
                ;Once started, Timer I must be cleared more often than once every 1024
                ;machine cycles. If Timer I is allowed to overflow, a Timer I
                ;interrupt will be generated. Thus, if global interrupts or the Timer
                ;I interrupt are inhibited, the watchdog function will be disabled.
                ;Also, if the watchdog interrupt occurs during another interrupt
                ;service, it will be delayed until an RETI (return from interrupt)
                ;instruction is executed. The I2C bus pins SCL and SDA may be used as
                ;open drain outputs.

                ;*******************************************************************************

                    $MOD751
                    $Title(Timer I Watchdog)
                    $Date(11-06-90)
                    $Debug

0000                          ORG     0
0000 0126                     AJMP    Reset

001B                          ORG     1Bh          ;Timer I interrupt.
001B C2AF           TimerI:   CLR     EA           ;Get here only if watchdog overflows.
001D C2DC                     CLR     TIRUN        ;Turn off Timer I.
001F D2DD                     SETB    CLRTI        ;Clear Timer I interrupt.
0021 1125                     ACALL   XRETI        ;Force interrupt pending to clear.
0023 0126                     AJMP    Reset        ;Do a warm start.
0025 32             XRETI:    RETI

0026 758107         Reset:    MOV     SP,#7h       ;Initialize the stack pointer.

                ;Note: it is important to force the stack pointer to a particular
                ;starting value in this application because we may be re-starting
                ;after a watchdog interrupt, with the stack in an unknown condition.

0029 75D800         MOV     I2CFG,#0               ;Initialize I2CFG (set up CT0, CT1).
002C D2DC           SETB    TIRUN                  ;Enable Timer I run.
002E D2AB           SETB    ETI                    ;Enable Timer I interrupt.
0030 D2AF           SETB    EA                     ;Enable interrupt system.


                ;The following is a "dummy" main program to test the watchdog timer.

0032 1153           Loop:     ACALL   Delay1       ;Wait 901 machine cycles.
0034 114E                     ACALL   WdRst        ;Reset Watchdog.
0036 1153                     ACALL   Delay1       ;Wait 901 machine cycles.
0038 114E                     ACALL   WdRst        ;Reset Watchdog.
003A 1153                     ACALL   Delay1       ;Wait 901 + 4 for ACALL & prior RET.
003C 1157                     ACALL   Delay2       ;Wait 108 + 2 for ACALL.
003E 00                       NOP     ;1016
003F 00                       NOP     ;1017
0040 00                       NOP     ;1018
0041 00                       NOP     ;1019
0042 00                       NOP     ;1020
```

```
TIWD                                Timer I Watchdog                                    11-06-90   PAGE 2

0043 00                     NOP     ;1021
0044 00                     NOP     ;1022
0045 00                     NOP     ;1023
0046 00                     NOP     ;1024           : Should get 'bitten' here.
0047 00                     NOP     ;1025
0048 00                     NOP     ;1026
0049 00                     NOP     ;1027
004A 00                     NOP     ;1028
004B 00                     NOP     ;1029
004C 0132                   AJMP    Loop            ;Should never get here.

004E C2DC           WdRst:  CLR     TIRUN           ;Reset Watchdog timer (Timer I).
0050 D2DC                   SETB    TIRUN
0052 22                     RET

0053 7480           Delay1: MOV     A,#128          ;Wait 901 machine cycles (1).
0055 8002                   SJMP    DLoop           ;(2)
0057 740F           Delay2: MOV     A,#15           ;Wait 108 machine cycles (1).
0059 A3             DLoop:  INC     DPTR            ;Delay = (ACC * 7) + 2 mach. cyc (2).
005A A3                     INC     DPTR            ;(2)
005B 14                     DEC     A               ;(1)
005C 70FB                   JNZ     Dloop           ;(2)
005E 22                     RET                     ;(2)

                            END

ASSEMBLY COMPLETE, 0 ERRORS FOUND
```

Timer I in non-I²C applications of the 83/87C751/752 microcontrollers

AN427

```
TIWD                                Timer I Watchdog                                11-06-90   PAGE 3

CLRTI. . . . . . . . . . . . . . . B ADDR  00DDH  PREDEFINED
DELAY1 . . . . . . . . . . . . . . C ADDR  0053H
DELAY2 . . . . . . . . . . . . . . C ADDR  0057H
DLOOP. . . . . . . . . . . . . . . C ADDR  0059H
EA . . . . . . . . . . . . . . . . B ADDR  00AFH  PREDEFINED
ETI. . . . . . . . . . . . . . . . B ADDR  00ABH  PREDEFINED
I2CFG. . . . . . . . . . . . . . . D ADDR  00D8H  PREDEFINED
LOOP . . . . . . . . . . . . . . . C ADDR  0032H
RESET. . . . . . . . . . . . . . . C ADDR  0026H
SP . . . . . . . . . . . . . . . . D ADDR  0081H  PREDEFINED
TIMERI . . . . . . . . . . . . . . C ADDR  001BH  NOT USED
TIRUN. . . . . . . . . . . . . . . B ADDR  00DCH  PREDEFINED
WDRST. . . . . . . . . . . . . . . C ADDR  004EH
XRETI. . . . . . . . . . . . . . . C ADDR  0025H
```

Using the ADC and PWM of the 83C752/87C752

AN428

The Signetics 83C752/87C752 is a single-chip control-oriented microcontroller. It is an 80C51 derivative, having the same basic architecture and powerful instruction set in a small 28-pin package. As "add-on" functions to a standard microcontroller, it offers an I^2C small area network port, a five-channel multiplexed 8-bit analog-to-digital converter (ADC), and a pulse width modulation (PWM) output. The part is essentially the popular 8XC751 with the addition of the ADC and the PWM output.

There are many control applications for which this microcontroller can provide an almost-complete, low-cost solution. The A/D converter can monitor analog voltages of up to five sources. The PWM output can be used to generate an analog control voltage with the addition of a simple integrator circuit. Another potential use for the PWM output is as a driver of power-switching circuits for DC motor speed control.

The analog-to-digital converter has 8-bit resolution, and the conversion takes 40 machine cycles. A multiplexer selects one out of five input pins. The operation of the A/D converter and the multiplexer is controlled by the ADCON register.

The repetition frequency of the PWM output pulses is determined by an 8-bit prescaler, programmed at register PWMP. The duty cycle of these pulses is determined by the contents of a compare register, PWM. In order to implement the pulse width modulator, the prescaler output drives an 8-bit counter. When the counter value matches the contents of the compare (PWM) register, the PWM output is set high, and when the counter reaches zero, the output is set low. The counter is modulo 255, so the duty cycle generated will be the PWM contents multiplied by 1/255.

The enclosed listing demonstrates usage of the A/D converter and the PWM. In order to communicate with the outside world, the program sends messages on a software-driven RS-232 port. The routines for sending messages via a software-controlled serial port can be quite useful, and for further discussion on those, please refer to Application Note 423: "Software Driven Serial Communication Routines for the 83C751 and 83C752 Microcontrollers."

Bit 5 of port 1 is used for the RS-232 communications, and in order to hook the microcontroller to a terminal, a buffer (e.g., MC1488) is needed. Timer 0 is used as the baud rate generator, where the timer value is defined by the symbol BaudVal. The programmed value will generate a 9600 baud rate with a 16MHz crystal.

The program, after initialization and sending a message to the terminal, scans all five A/D channel inputs and outputs the voltage read on the serial port, as a hexadecimal value. Circuit operation can be verified by comparing channel voltages with the reading at the terminal. The program follows with an infinite loop in which channel 0 of the A/D converter is read, and its value is used to program the PWM. A simple verification of the duty cycle can be done with a voltmeter: since it acts as an integrator, its reading will be proportional to the duty cycle. Reading of a voltmeter on the PWM output should be proportional to the channel 0 input voltage. If the analog reference voltage AV_{CC}, which is full-scale of the A/D measurement, is set to be exactly as V_{CC}, the PWM output will track channel 0 within about 20mV.

Using the ADC and PWM of the 83C752/87C752

AN428

```
DEMO752C                    87C752 A/D and PWM Demonstration                    12/03/90    PAGE 1

                 ;

                 ;*************************************************************************

                 ;                  87C752 A/D and PWM Demonstration Program

                 ;  This program first reads all five A/D channels and outputs the values in
                 ;  hexadecimal as RS-232 data. Next, the PWM output is set to reflect the
                 ;  value on A/D channel 0, and again outputs the A/D value to RS-232. Note
                 ;  that the A/D value is inverted before being moved to the PWM compare
                 ;  register in order to compensate for the inversion on the PWM output pin.
                 ;  This process is repeated continuously.

                 ;  Thus, a voltage may be applied to ADC0 (P1.0, pin 13) to vary the PWM pulse
                 ;  width.  A simple test of this function is to measure the voltage on ADC0
                 ;  and PWM with a voltmeter.  A typical voltmeter will integrate the waveform
                 ;  on the PWM output and show a voltage within about 20mV of that on ADC0.

                 ;  The RS-232 output appears on Port 1 pin 5, which must be buffered with an
                 ;  MC1488 or perhaps a MAX232 chip prior to being connected to a terminal.
                 ;  The transmission rate will  be 9600 baud when the 87C752 is operated from
                 ;  16MHz crystal.

                 ;*************************************************************************

                 $Title(87C752 A/D and PWM Demonstration)
                 $Date(12/03/90)
                 $MOD752

                 ;*************************************************************************

FF75             BaudVal  EQU      -139              ;Timer value for 9600 baud @ 16 MHz.
                                                     ;(one bit cell time)

0010             XmtDat   DATA     10h               ;Data for RS-232 transmit routine.
0012             BitCnt   DATA     12h               ;RS-232 transmit bit count.
0013             PWMVal   DATA     13h               ;Holds next value for updating the PWM.
0014             ADVal    DATA     14h               ;Holds last A/D conversion result.

0020             Flags    DATA     20h
0000             TxFlag   BIT      Flags.0           ;Transmit-in-progress flag.
0001             ADFlag   BIT      Flags.1           ;Indicates A/D conversion complete.

0095             TxD      BIT      P1.5              ;Port bit for RS-232 transmit.

                 ;*************************************************************************

                 ; Interrupt Vectors

0000                      ORG      0                 ;Reset vector.
0000 0135                 AJMP     Reset

000B                      ORG      0BH               ;Timer 0 interrupt.
000B 01C5                 AJMP     Timr0             ;(used as a baud rate generator)

002B                      ORG      2Bh               ;A/D conversion complete interrupt.
002B 0199                 AJMP     ADInt
```

```
DEMO752C                    87C752 A/D and PWM Demonstration                    12/03/90    PAGE 2

0033                            ORG     33h                 ;PWM interrupt.
0033 01A3                       AJMP    PWMInt


                    ;*****************************************************************************

0035 758130         Reset:      MOV     SP,#30h
0038 752000                     MOV     Flags,#0            ;Clear RS-232 flags.
003B 758800                     MOV     TCON,#00h           ;Set up timer controls.
003E 75A882                     MOV     IE,#82h             ;Enable timer 0 interrupt.

0041 90011B                     MOV     DPTR,#Msg1          ;Point to message string.
0044 310A                       ACALL   Mess                ;Send message.

0046 7900                       MOV     R1,#0               ;Start with A/D channel 0.
0048 E9             Loop1:      MOV     A,R1
0049 118D                       ACALL   ADConv              ;Start A/D conversion.
004B FA                         MOV     R2,A

004C 900152                     MOV     DPTR,#Msg2          ;Point to message string.
004F 310A                       ACALL   Mess                ;Send message.
0051 E9                         MOV     A,R1
0052 11EC                       ACALL   PrByte              ;Print channel #.
0054 900161                     MOV     DPTR,#Msg3          ;Point to message string.
0057 310A                       ACALL   Mess                ;Send message.

0059 EA                         MOV     A,R2
005A 11EC                       ACALL   PrByte              ;Print A/D value.
005C 09                         INC     R1                  ;Advance R1 value.
005D B905E8                     CJNE    R1,#5,Loop1
0060 90014F                     MOV     DPTR,#CRLF          ;Point to message string.
0063 310A                       ACALL   Mess

                    ; Now use A/D channel 0 value to control the PWM.

0065 758FFF                     MOV     PWMP,#0FFh          ;Set PWM slow frequency.
0068 758E00                     MOV     PWCM,#0             ;Set initial PWM value.
006B 751300                     MOV     PWMVal,#0           ;Default starting value for the PWM.
006E 75FE01                     MOV     PWENA,#1            ;Start PWM
0071 75A8CA                     MOV     IE,#0CAh            ;Now enable the A/D and PWM interrupts.

0074 7400           Loop2:      MOV     A,#0                ;Read A/D channel 0.
0076 1186                       ACALL   ADStart             ;Start A/D conversion.
0078 3001FD                     JNB     ADFlag,$            ;Wait for A/D conversion complete.
007B E514                       MOV     A,ADVal             ;Get A/D result to print.
007D 11EC                       ACALL   PrByte              ;Print PWM value.
007F 900165                     MOV     DPTR,#Msg4          ;Point to message string.
0082 310A                       ACALL   Mess
0084 80EE                       SJMP    Loop2


                    ; A/D Conversion Routines.
                    ;   The following shows two ways to use the A/D. Both routines are used by
                    ;   different portions of the sample program.

                    ;   Method 1: This version of the routine starts the conversion and then
                    ;   returns. The mainline program can detect when the conversion is
                    ;   complete by checking the A/D conversion complete flag (ADFlag) which is
```

```
DEMO752C                    87C752 A/D and PWM Demonstration                    12/03/90    PAGE 3

                    ;   set by the A/D interrupt service routine. A/D data must be read by the
                    ;   calling routine.

0086 C201           ADStart: CLR      ADFlag              ;Clear A/D conversion complete flag.
0088 4428                    ORL      A,#28h              ;Add control bits to channel #.
008A F5A0                    MOV      ADCON,A             ;Start conversion.
008C 22                      RET


                    ;   Method 2: This is an alternative version of the A/D routine which
                    ;   starts the conversion and then waits for it to complete before
                    ;   returning. A/D data is returned in the ACC.

008D 4428           ADConv:  ORL      A,#28h              ;Add control bits to channel #.
008F F5A0                    MOV      ADCON,A             ;Start conversion.
0091 E5A0           ADC1:    MOV      A,ADCON
0093 30E4FB                  JNB      ACC.4,ADC1          ;Wait for conversion complete.
0096 E584                    MOV      A,ADAT              ;Read A/D.
0098 22                      RET


                    ; A/D interrupt service routine.

0099 E584           ADInt:   MOV      A,ADAT              ;Read A/D data.
009B F514                    MOV      ADVal,A             ;Save A/D data for print routine.
009D F4                      CPL      A                   ;Complement the value for the PWM.
009E F513                    MOV      PWMVal,A            ;Set new value for PWM update.
00A0 D201                    SETB     ADFlag              ;Tell main that new A/D data is ready.
00A2 32                      RETI


                    ; PWM interrupt service routine allows updating the PWM synchronously.

00A3 85138E         PWMInt:  MOV      PWCM,PWMVal         ;Update PWM duty cycle.
00A6 32                      RETI


                    ; Send a byte out RS-232 and wait for completion before returning.

00A7 11AD           XmtByte: ACALL    RSXmt               ;Send ACC to RS-232 output.
00A9 2000FD                  JB       TxFlag,$            ;Wait for transmit complete.
00AC 22                      RET


                    ; Begin RS-232 transmit.

00AD F510           RSXmt:   MOV      XmtDat,A            ;Save data to be transmitted.
00AF 75120A                  MOV      BitCnt,#10          ;Set bit count.
00B2 758CFF                  MOV      TH,#High BaudVal    ;Set timer for baud rate.
00B5 758A75                  MOV      TL,#Low BaudVal
00B8 758DFF                  MOV      RTH,#High BaudVal   ;Also set timer reload value.
00BB 758B75                  MOV      RTL,#Low BaudVal
00BE D28C                    SETB     TR                  ;Start timer.
00C0 C295                    CLR      TxD                 ;Begin start bit.
00C2 D200                    SETB     TxFlag              ;Set transmit-in-progress flag.
00C4 22                      RET
```

```
DEMO752C                    87C752 A/D and PWM Demonstration                    12/03/90    PAGE 4

                    ; Timer 0 timeout: RS-232 receive bit or transmit bit.

00C5 C0E0           Timr0:   PUSH     ACC
00C7 C0D0                    PUSH     PSW
00C9 200007                  JB       TxFlag,TxBit        ;Is this a transmit timer interrupt?
00CC C28C           T0Ex1:   CLR      TR                  ;Stop timer.
00CE D0D0           T0Ex2:   POP      PSW
00D0 D0E0                    POP      ACC
00D2 32                      RETI


                    ; RS-232 transmit bit routine.

00D3 D51204         TxBit:   DJNZ     BitCnt,TxBusy       ;Decrement bit count, test for done.
00D6 C200                    CLR      TxFlag              ;End of stop bit, release timer.
00D8 80F2                    SJMP     T0Ex1               ;Stop timer and exit.

00DA E512           TxBusy:  MOV      A,BitCnt            ;Get bit count.
00DC B40104                  CJNE     A,#1,TxNext         ;Is this a stop bit?
00DF D295                    SETB     TxD                 ;Set stop bit.
00E1 80EB                    SJMP     T0Ex2               ;Exit.

00E3 E510           TxNext:  MOV      A,XmtDat            ;Get data.
00E5 13                      RRC      A                   ;Advance to next bit.
00E6 F510                    MOV      XmtDat,A
00E8 9295                    MOV      TxD,C               ;Send data bit.
00EA 80E2                    SJMP     T0Ex2               ;Exit.


                    ; Print byte routine: print ACC contents as ASCII hexadecimal.

00EC C0E0           PrByte:  PUSH   ACC
00EE C4                      SWAP   A
00EF 11FA                    ACALL  HexAsc
00F1 11A7                    ACALL  XmtByte
00F3 D0E0                    POP    ACC
00F5 11FA                    ACALL  HexAsc            ;Print nibble in ACC as ASCII hex.
00F7 11A7                    ACALL  XmtByte
00F9 22                      RET


                    ; Hexadecimal to ASCII conversion routine.

00FA 540F           HexAsc:  ANL    A,#0FH            ;Convert a nibble to ASCII hex.
00FC 30E308                  JNB    ACC.3,NoAdj
00FF 20E203                  JB     ACC.2,Adj
0102 30E102                  JNB    ACC.1,NoAdj
0105 2407           Adj:     ADD    A,#07H
0107 2430           NoAdj:   ADD    A,#30H
0109 22                      RET


                    ; Message string transmit routine.

010A C0E0           Mess:    PUSH     ACC
010C 7800                    MOV      R0,#0               ;R0 is character pointer (string
010E E8             Mesl:    MOV      A,R0                ; length is limited to 256 bytes).
010F 93                      MOVC     A,@A+DPTR           ;Get byte to send.
```

```
DEMO752C                    87C752 A/D and PWM Demonstration                    12/03/90    PAGE 5

0110 B40003         233             CJNE    A,#0,Send           ;End of string is indicated by a 0.
0113 D0E0           234             POP     ACC
0115 22             235             RET
                    236
0116 11A7           237     Send:   ACALL   XmtByte              ;Send a character.
0118 08             238             INC     R0                  ;Next character.
0119 80F3           239             SJMP    Mes1
                    240
011B 0D0A           241     Msg1:   DB      0Dh, 0Ah,
011D 54686973       242             DB      'This is a demonstration of the 87C752 A/D and PWM.'
0121 20697320
0125 61206465
0129 6D6F6E73
012D 74726174
0131 696F6E20
0135 6F662074
0139 68652038
013D 37433735
0141 3220412F
0145 4420616E
0149 64205057
014D 4D2E
014F 0D0A00         243     CRLF:   DB      0Dh, 0Ah, 0
                    244
0152 0D0A412F       245     Msg2:   DB      0Dh, 0Ah, 'A/D Channel ', 0
0156 44204368
015A 616E6E65
015E 6C2000
                    246
0161 203D2000       247     Msg3:   DB      ' = ', 0
                    248
0165 202000         249     Msg4:   DB      '  ', 0
                    250
                    251             END

ASSEMBLY COMPLETE, 0 ERRORS FOUND
```

```
DEMO752C                    87C752 A/D and PWM Demonstration                    12/03/90    PAGE 6

ACC. . . . . . . . . . . . . . . . D ADDR  00E0H  PREDEFINED
ADAT . . . . . . . . . . . . . . . D ADDR  0084H  PREDEFINED
ADC1 . . . . . . . . . . . . . . . C ADDR  0091H
ADCON. . . . . . . . . . . . . . . D ADDR  00A0H  PREDEFINED
ADCONV . . . . . . . . . . . . . . C ADDR  008DH
ADFLAG . . . . . . . . . . . . . . B ADDR  0001H
ADINT. . . . . . . . . . . . . . . C ADDR  0099H
ADJ. . . . . . . . . . . . . . . . C ADDR  0105H
ADSTART. . . . . . . . . . . . . . C ADDR  0086H
ADVAL. . . . . . . . . . . . . . . D ADDR  0014H
BAUDVAL. . . . . . . . . . . . . .   NUMB  FF75H
BITCNT . . . . . . . . . . . . . . D ADDR  0012H
CRLF . . . . . . . . . . . . . . . C ADDR  014FH
FLAGS. . . . . . . . . . . . . . . D ADDR  0020H
HEXASC . . . . . . . . . . . . . . C ADDR  00FAH
IE . . . . . . . . . . . . . . . . D ADDR  00A8H  PREDEFINED
LOOP1. . . . . . . . . . . . . . . C ADDR  0048H
LOOP2. . . . . . . . . . . . . . . C ADDR  0074H
MESL . . . . . . . . . . . . . . . C ADDR  010EH
MESS . . . . . . . . . . . . . . . C ADDR  010AH
MSG1 . . . . . . . . . . . . . . . C ADDR  011BH
MSG2 . . . . . . . . . . . . . . . C ADDR  0152H
MSG3 . . . . . . . . . . . . . . . C ADDR  0161H
MSG4 . . . . . . . . . . . . . . . C ADDR  0165H
NOADJ. . . . . . . . . . . . . . . C ADDR  0107H
P1 . . . . . . . . . . . . . . . . D ADDR  0090H  PREDEFINED
PRBYTE . . . . . . . . . . . . . . C ADDR  00ECH
PSW. . . . . . . . . . . . . . . . D ADDR  00D0H  PREDEFINED
PWCM . . . . . . . . . . . . . . . D ADDR  008EH  PREDEFINED
PWENA. . . . . . . . . . . . . . . D ADDR  00FEH  PREDEFINED
PWMINT . . . . . . . . . . . . . . C ADDR  00A3H
PWMP . . . . . . . . . . . . . . . D ADDR  008FH  PREDEFINED
PWMVAL . . . . . . . . . . . . . . D ADDR  0013H
RESET. . . . . . . . . . . . . . . C ADDR  0035H
RSXMT. . . . . . . . . . . . . . . C ADDR  00ADH
RTH. . . . . . . . . . . . . . . . D ADDR  008DH  PREDEFINED
RTL. . . . . . . . . . . . . . . . D ADDR  008BH  PREDEFINED
SEND . . . . . . . . . . . . . . . C ADDR  0116H
SP . . . . . . . . . . . . . . . . D ADDR  0081H  PREDEFINED
T0EX1. . . . . . . . . . . . . . . C ADDR  00CCH
T0EX2. . . . . . . . . . . . . . . C ADDR  00CEH
TCON . . . . . . . . . . . . . . . D ADDR  0088H  PREDEFINED
TH . . . . . . . . . . . . . . . . D ADDR  008CH  PREDEFINED
TIMR0. . . . . . . . . . . . . . . C ADDR  00C5H
TL . . . . . . . . . . . . . . . . D ADDR  008AH  PREDEFINED
TR . . . . . . . . . . . . . . . . B ADDR  008CH  PREDEFINED
TXBIT. . . . . . . . . . . . . . . C ADDR  00D3H
TXBUSY . . . . . . . . . . . . . . C ADDR  00DAH
TXD. . . . . . . . . . . . . . . . B ADDR  0095H
TXFLAG . . . . . . . . . . . . . . B ADDR  0000H
TXNEXT . . . . . . . . . . . . . . C ADDR  00E3H
XMTBYTE. . . . . . . . . . . . . . C ADDR  00A7H
XMTDAT . . . . . . . . . . . . . . D ADDR  0010H
```

Airflow measurement using the 83/87C752 and "C"

AN429

INTRODUCTION

This application note describes a low-cost airflow measurement device based on the Signetics 83/87C752 microcontroller. Airflow measurement—determining the volume of air transferred per unit time (cubic feet per minute, or cfm)—is intrinsic to a variety of industrial and scientific processes.

Airflow computation depends on three simultaneous physical air measurements—velocity, pressure, and temperature. This design includes circuits and sensors allowing the 8XC752 to measure all three parameters.

The design also includes seven-segment LED displays, discrete LEDs, and pushbutton switches to allow selective display of airflow, temperature, and pressure. Furthermore, airflow is continuously compared with a programmer-defined setpoint. Should the measured airflow exceed the setpoint, an output relay is energized. In actual application, this relay output could be used to signal the setpoint violation (via lamp or audio annunciator) or otherwise control the overall process (e.g., emergency process shutdown). Of course, the setpoint, comparison criteria (greater, less than, etc.) and violation response (relay on, relay off) are easily changed by program modification to meet actual application requirements.

Referring to Figure 1, the overall operation of the airflow device is as follows.

Normally the unit continuously displays the airflow (in cfm) on the seven-segment displays. The discrete CFM LED is also lit to confirm the parameter being displayed.

Pressing the TEMP pushbutton switches the display to temperature (in degrees C) and lights the TEMP LED. As long as the pushbutton remains pressed, the temperature is displayed. When the pushbutton is released, the display reverts to the default pressure display.

Similarly, pressing the PSI pushbutton displays the atmospheric pressure (in pounds per square inch) and lights the PSI LED. The pressure is displayed as long as the pushbutton is pressed, and the default airflow display resumes when the pushbutton is released.

Finally, pressing the SET-POINT pushbutton displays the programmed airflow setpoint (in cfm) and lights the SET-POINT LED. Again, releasing the pushbutton causes the display to revert to the default airflow measurement.

CONTROL PROGRAMMING IN "C"

While, thanks to advanced semiconductor processing, hardware price/performance continues to improve, software development technology has changed little over time. Thus, given ever-rising costs for qualified personnel, software "productivity" is arguably in decline. Indeed, for low-unit cost and/or low-volume applications, software development has emerged as the major portion of total design cost. Furthermore, beyond the initial programming cost, "hidden" costs also arise in the form of life-cycle code maintenance and revision and lost revenue/market share due to excessive time-to-market.

Traditionally, control applications have been programmed in assembly language to overcome microcontroller resource and performance constraints. Now, thanks to more powerful microcontrollers and advanced compiler technology, it is feasible to program control applications using a High-Level Language (HLL).

The primary benefit of using an HLL is obvious—one HLL program "statement" can perform the same function as many lines of assembly language. Furthermore, a well-written HLL program will typically be more "readable" than an assembly language equivalent, resulting in reduced maintenance and revision/upgrade costs.

Of the many popular HLLs, the "C" language has emerged as the major contender for control applications. More than other languages, C gives the programmer direct access to, and control of, low-level hardware resources—a requirement for deterministic, real-time I/O applications. Furthermore, C is based on a "minimalist" philosophy in which the language performs only those functions explicitly requested by the programmer. This approach is well-suited for control applications, which are often characterized by strict cost and performance requirements.

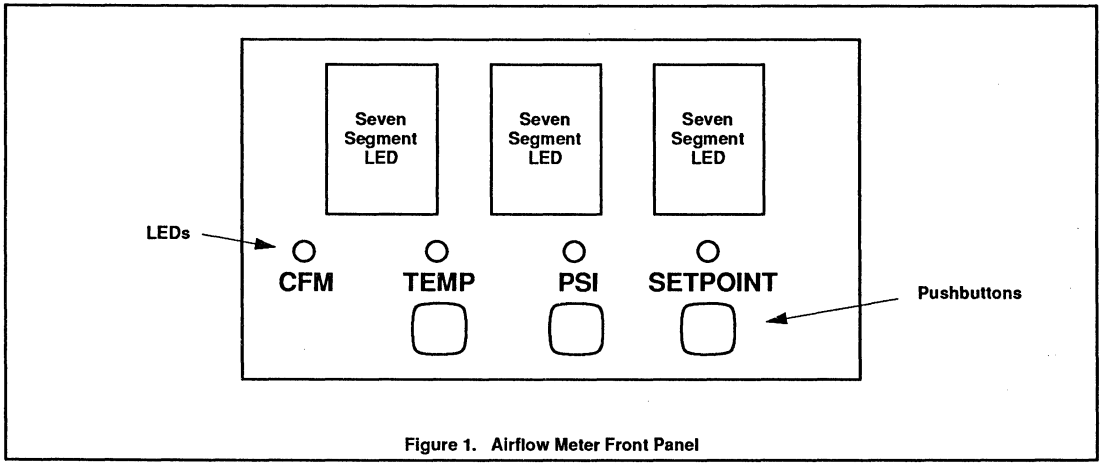


Figure 1. Airflow Meter Front Panel

Airflow measurement using the 83/87C752 and "C"

AN429

8XC752 OVERVIEW

The 83C752/87C752 (ROM/EPROM-based) combine the performance advantages of the 8-bit 80C51 architecture with the low cost, power consumption, and size/pin count of a 4-bit microcontroller. Therefore, the 8XC752 is uniquely capable of bringing high processing speed and HLL programming to even the most cost-sensitive applications such as handheld (battery driven) instruments, automotive distributed processing, "smart" appliances, and sophisticated consumer electronics.

Obviously, the 8XC752 can be used for cost-reduced versions of existing 8-bit applications. The device can also replace similarly priced 4-bit devices to achieve benefits of higher performance and, most importantly, easier s/w development including the use of HLL. Indeed, the component and system design costs associated with the 8XC752 are so low that it is a viable candidate for first-time computerization of formerly non-microcontroller-based designs.

Figure 2 shows the block diagram of the 8XC752. Major features of the device include the following.

Full-Function, High-Speed (to 16MHz) 80C51 CPU Core

The popular 80C51 architecture features 8- and 16-bit processing and high-speed execution. Most instructions execute in a single machine cycle (the slowest instructions require only two cycles). Though a streamlined architecture, the CPU core, unlike 4-bit devices, includes all the basic capabilities (such as stack, multiply instruction, interrupts, etc.) required to support HLL compilation. The CPU core also includes a unique Boolean processor which is well-suited for the bit-level processing and I/O common to control applications.

Low-Power CMOS and Power-Saving Operation Modes

Thanks to the advanced CMOS process, the 8XC752 features extremely low power consumption, which helps to extend battery life in handheld applications and otherwise reduce power supply and thermal dissipation costs and reliability concerns. Low ACTIVE mode (full-speed operation) power consumption—only 11mA typical at 12MHz—is further complemented by two program-initiated power-saving operation modes—IDLE and POWER-DOWN.

In idle mode, CPU instruction processing stops while on-chip I/O and RAM remain powered. Power consumption drops to 1.5μA (typical, 12MHz) until processing is restarted by interrupt or reset. Power-down mode cuts power consumption further (to only 10μA typical at 12MHz) by stopping both instruction and I/O processing. Return to full-speed operation from power-down mode is via reset.

Note that power consumption can be further cut by reducing the clock frequency as much as application performance requirements allow, as shown in Figure 3.

Another virtue of the CMOS process is superior tolerance to variations in V_{CC}, a requirement inherent in the targeted applications. The EPROM-based device (87C752) operates over a V_{CC} range of 4.5V to 5.5V, while the ROM-based device (83C752) will operate from 4V to 6V.

On-Chip ROM (83C752), EPROM (87C752), and RAM

The 8XC752 integrates 2048 bytes of program ROM/EPROM and 64 bytes of data RAM. This relatively small amount of memory reflects the fact that the targeted applications, though they may require high-speed processing, are typically characterized by simple algorithms and data structures. High code efficiency of the architecture means even this small amount of memory can effectively support the use of C. If necessary, the judicious use of assembly language can help bypass code size (and performance) constraints.

Five-Channel 8-Bit A/D Converter

Most control applications are characterized by the need to monitor "real-world" (i.e., analog) parameters. To this end, the 8XC752 includes a medium-speed (40 clock cycle conversion) 8-bit analog-to-digital (A/D) converter. Five separate input lines are provided along with multiplexer logic to select an input for conversion. The A/D converters speed, resolution, and accuracy are more than adequate to measure temperature, pressure, and other common environmental parameters.

Airflow measurement using the 83/87C752 and "C"

AN429

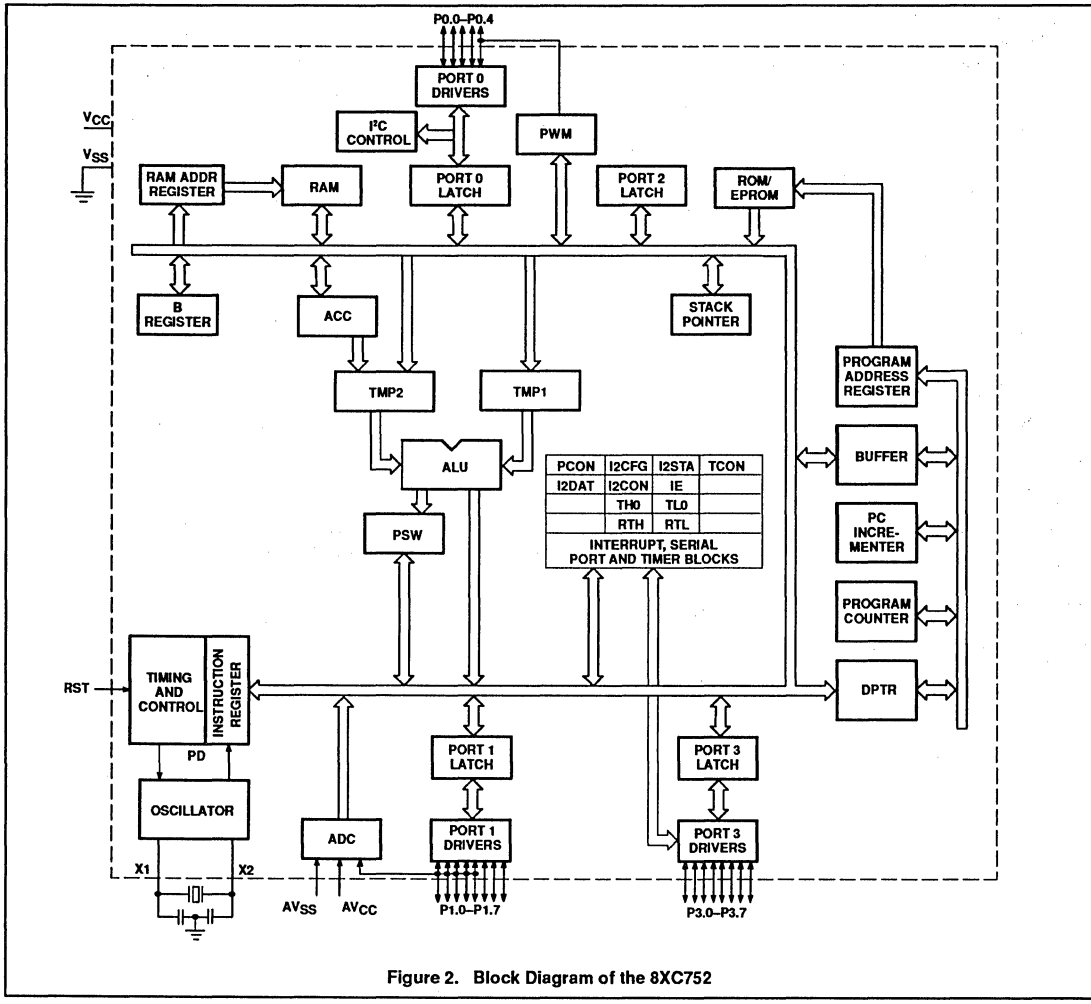


Figure 2. Block Diagram of the 8XC752

Timer/Counters

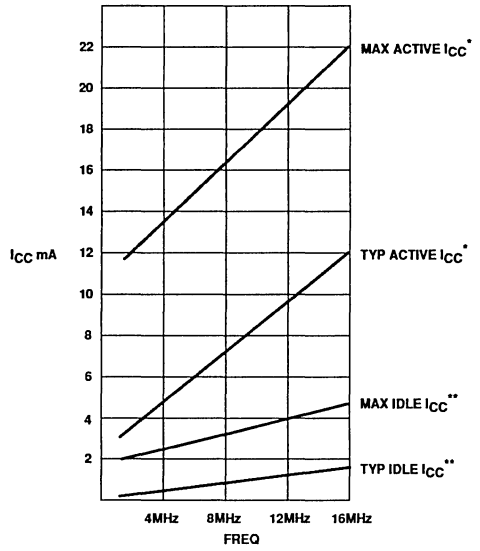
Control applications, due to their "real-time" nature, invariably call for a variety of timing and counting capabilities. The 8XC752 meets the need by integrating three separate functions—a 16-bit auto-reload counter/timer, an 8-bit pulse width modulator (PWM) output/timer, and a fixed-rate timer for timebase generation. Together, these timing/counting resources can serve a range of tasks, including waveform generation, external event counting, elapsed time calculation, periodic interrupt generation, and watchdog timer.

I²C Bus

The Inter-Integrated Circuit (I²C) bus is a patented serial peripheral interface. The virtue of I²C is the ability to expand system functionality with acceptable performance and minimum cost. Notably, the pin and interconnect count is radically reduced compared to expansion via a typical microprocessor bus—I²C requires only two lines, while a parallel bus often consumes 20–30 lines and may call for extra glue logic (decoder, address latch, etc.). The 8XC752 I²C port allows easy connection to a wide variety of compatible peripherals such as LCD drivers, A/D and D/A converters, consumer/telecom and special-purpose memory (e.g., EEPROM). I²C can also be used to build distributed processing systems connecting multiple I²C-compatible microcontrollers.

Airflow measurement using the 83/87C752 and "C"

AN429



NOTES:
Maximum I_{CC} values taken at V_{CC} = 5.5V and worst case temperature. Typical I_{CC} values taken at V_{CC} = 5.0V and 25°C.
* I_{CC} is measured with all output pins disconnected; X1 driven with t_{CLCH}, t_{CHCL} = 5ns, V_{IL} = V_{SS} + 0.5V, V_{IH} = V_{CC} – 0.5V; X2 n.c.; RST = port 0 = V_{CC}. I_{CC} will be slightly higher if a crystal oscillator is used. (This refers to AC Electrical Characteristics.)
** Idle I_{CC} is measured with all output pins disconnected; X1 driven with t_{CLCH}, t_{CHCL} = 5ns, V_{IL} = V_{SS} + 0.5V, V_{IH} = V_{CC} – 0.5V; X2 n.c.; port 0 = V_{CC}; RST = V_{SS}. (This refers to AC Electrical Characteristics.)

Figure 3. I_{CC} vs. FREQ

8XC752 PIN FUNCTIONS

Since the 8XC752 is packaged in a cost/space-saving 28-pin package DIP or PLCC), a flexible mapping of I/O functions to pins is required to ensure the widest possible application coverage.

Of the 28 pins, seven pins are allocated to basic functions, including digital power (V_{CC}, V_{SS}), analog reference (AV_{CC}, AV_{SS}), clock oscillator (X1, X2), and reset (RST). Thus, 21 pins, organized into three ports (5-bit port 0, 8-bit ports 1 and 3), are available for user I/O.

Figure 4 shows the alternative uses for these 21 lines. As shown, the mapping is quite versatile, which maximizes the access to on-chip I/O functions and helps ensure full pin utilization.

P0.0 TTL IN/OUT (open drain), I^2C clock (SCLK)
P0.1 TTL IN/OUT (open drain), I^2C data (SDA)
P0.2 TTL IN/OUT (open drain)
P0.3 TTL IN/OUT (internal pull-up)
P0.4 TTL IN/OUT (internal pull-up), PWM output

P1.0 TTL IN/OUT (internal pull-up), A/D input channel 0
P1.1 TTL IN/OUT (internal pull-up), A/D input channel 1
P1.2 TTL IN/OUT (internal pull-up), A/D input channel 2
P1.3 TTL IN/OUT (internal pull-up), A/D input channel 3
P1.4 TTL IN/OUT (internal pull-up), A/D input channel 4
P1.5 TTL IN/OUT (internal pull-up), INT0 interrupt input
P1.6 TTL IN/OUT (internal pull-up), INT1 interrupt input
P1.7 TTL IN/OUT (internal pull-up), TIMER 0 (T0) input

NOTE: P1.0–P1.4 may only be changed as a group, i.e., either all TTL I/O or all A/D inputs. However, when selected as A/D inputs, P1.0–P1.4 may also be used as TTL inputs.

P3.0 TTL IN/OUT (internal pull-up)
P3.1 TTL IN/OUT (internal pull-up)
P3.2 TTL IN/OUT (internal pull-up)
P3.3 TTL IN/OUT (internal pull-up)
P3.4 TTL IN/OUT (internal pull-up)
P3.5 TTL IN/OUT (internal pull-up)
P3.6 TTL IN/OUT (internal pull-up)
P3.7 TTL IN/OUT (internal pull-up)

Figure 4. 8XC752 I/O Port Description

Airflow measurement using the 83/87C752 and "C"

AN429

AIRFLOW METER CIRCUIT DESCRIPTION

Figure 5 is the schematic diagram of the airflow meter circuit. As shown, the 8XC752 is connected to the following function blocks.

Discrete and Seven-Segment LED Display

The seven-segment LEDs display the parameter of interest (airflow, temperature, pressure, or setpoint). A discrete LED associated with each parameter is lit when that parameter is being displayed.

The seven-segment LEDs are identified as X0.1, X1, and X10, reflecting their decimal position (tenths, ones, and tens, respectively). Each display has eight data inputs (the seven segments and a decimal point) and common terminals which allow the display to be enabled or blanked. The eight data inputs, and the four discrete LEDs, are driven from port 3 of the 8XC752 via high-current driver U2 and current limiting resistors RP1.

Since all the segmented and discrete LEDs share common data lines, data display must be time multiplexed. Transistors Q1–Q4 connect to separate output lines of port 0, allowing a particular seven-segment LED or the discrete LEDs (as a group) to be individually enabled for display. This type of LED multiplexing is quite common since, at a fast enough refresh rate, the switching between displays is not perceptible by the operator. The major benefit is the reduction of I/O lines required (without multiplexing, 28, rather than 8, data lines would be required).

Pushbutton Switch Inputs

Three pushbuttons select the parameter to be displayed—temperature, pressure, or setpoint (when no button is pressed, airflow is displayed). The four states (SW1, SW2, SW3, or no button pressed) are effectively encoded onto two port 1 input lines (taking advantage of the capability to use port 1 lines configured as A/D for TTL input) as follows:

	P1.3	P1.4
No button pressed	HIGH	HIGH
SW1 (TEMP) pressed	LOW	HIGH
SW2 (PSI) pressed	HIGH	LOW
SW3 (SETPOINT) pressed	LOW	LOW

The only impact of this encoding scheme is that SW3 has a higher priority than the other pushbuttons—a factor of no concern in this simple application. Similarly, latching, debouncing, rollover, or other conditioning of the pushbutton inputs is not required.

Setpoint Control

This is simply a variable resistor voltage divider which serves to establish an analog voltage corresponding to an airflow threshold at which action is taken. It connects to a port 1 A/D input.

Relay Output

When an airflow setpoint violation is detected, DPDT relay K1 is energized via P1.6, which is configured as a TTL output, buffered by transistor Q5.

Flowmeter Input

Measurement of the air velocity is via an air turbine tachometer connected, via optoisolator U7, to P1.5, which is configured as a TTL input. The tachometer input is assumed to be a negative-going pulse train with less than 10% duty cycle.

Air Pressure Sensor

To determine airflow, the air velocity must be factored by ambient pressure—for a given velocity (and temperature), lower/higher atmospheric pressure will correspond with lower/higher airflow. The pressure sensor, U3, outputs a voltage differential corresponding to the pressure. Amplifier U4 conditions the pressure sensor output to the range of AV_{SS} to AV_{CC} (the analog references for the 8XC752 A/D converter). The conditioned pressure sensor output is presented to A/D input P1.0.

To calibrate the pressure sensor, press the PSI pushbutton and adjust the gain pot (R1) until the display matches the local atmospheric pressure in pounds per square inch (14.7 at sea level).

Air Temperature Sensor

Similar to pressure, ambient temperature also affects the airflow calculation. For a given air velocity (and pressure), higher/lower temperature will correspond with lower/higher airflow. Temperature sensor U5 outputs an absolute voltage corresponding to temperature. Amplifier U6 conditions the temperature sensor output to the range AV_{SS} to AV_{CC} for connection to A/D input P1.1.

To calibrate the temperature sensor, adjust the gain pot (R5) so that the display (while pressing the TEMP pushbutton) matches the measured output of U5 (LM35).

Figure 6 summarizes the usage of the 8XC752 I/O lines in this application.

Airflow measurement using the 83/87C752 and "C"

AN429

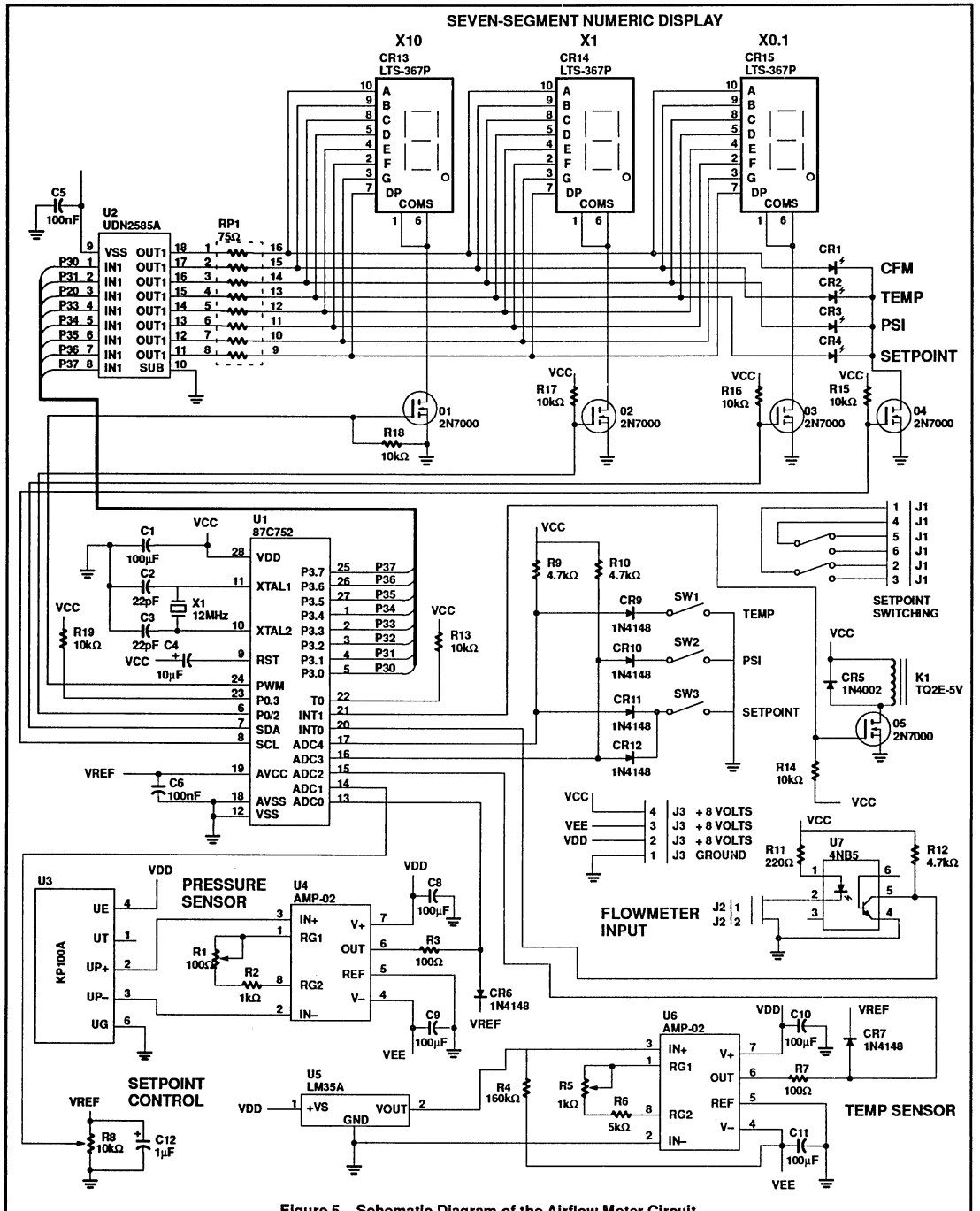


Figure 5. Schematic Diagram of the Airflow Meter Circuit

Airflow measurement using the 83/87C752 and "C"

P0.0 TTL OUT—Enables the discrete LEDs
P0.1 TTL OUT—Enables the tenths digit seven-segment LED
P0.2 TTL OUT—Enables the ones digit seven-segment LED
P0.3 Pulled up
P0.4 TTL OUT—Enables the tens digit seven-segment LED

P1.0 A/D input—Connected to analog air pressure sensor
P1.1 A/D input—Connected to analog air temperature sensor
P1.2 A/D input—Connected to analog setpoint control
P1.3 TTL IN—One of two pushbutton input lines
P1.4 TTL IN—The second pushbutton input line
P1.5 INT0 interrupt input—Air turbine tachometer input
P1.6 TTL OUT—Setpoint relay control
P1.7 Pulled up

NOTE: P1.0–P1.4 may only be changed as a group, i.e., either all TTL I/O or all A/D inputs. However, when selected as A/D inputs, P1.0–P1.4 may also be used as TTL inputs.

P3.0 TTL OUT—Seven-segment LEDs segment A, CFM discrete LED
P3.1 TTL OUT—Seven-segment LEDs segment B, TEMP discrete LED
P3.2 TTL OUT—Seven-segment LEDs segment C, PSI discrete LED
P3.3 TTL OUT—Seven segment LEDs segment D, SETPOINT discrete LED
P3.4 TTL OUT—Seven segment LEDs segment E
P3.5 TTL OUT—Seven segment LEDs segment F
P3.6 TTL OUT—Seven segment LEDs segment G
P3.7 TTL OUT—Seven segment LEDs segment DP

Figure 6. Airflow Meter I/O Port Usage

SOFTWARE DEVELOPMENT PROCEDURE

The airflow meter application software is almost entirely written in C using a development package from Franklin Software. The Franklin Software C compiler is a cross-compiler that runs on the IBM PC (and compatibles) while generating code suitable for execution by any 80C51-based product, including the 8XC752. For more information, contact:

Franklin Software
888 Saratoga Ave., #2
San Jose, CA 95129

The process of developing a C program using the Franklin package (the process is similar for other third-party cross-compilers) is as follows:

1. The program is entered/edited on the PC using the programmer's preferred text editor.
2. The program is compiled on the PC with the Franklin C compiler.
3. Should compile errors (also known as syntax errors) occur, they are corrected by returning to step 1 until an error-free compile is achieved.
4. Before testing the compiled program, it needs to be combined, using the Franklin-supplied linker, with any required assembly language routines. Besides routines explicitly written by the programmer, every Franklin C program requires an assembly language startup routine (supplied by Franklin and, if necessary, edited by the programmer) which performs basic reset initialization and configuration operations before transferring control to the C program.
5. The compiled object code is tested for correct operation. This can either be accomplished by using an 80C51-family simulator running on the PC or by downloading the object code to an in-circuit emulator. The simulator approach has the virtues of low cost and consolidation of all work on the PC at the cost of non-real-time operation/debug constraints (the simulator may execute 100–1000 times slower than the microcontroller). The in-circuit emulator provides real-time operation and the additional benefit of assisting hardware design debug at somewhat higher cost.
6. Should program execution prove faulty (known as semantic errors), return to step 1 until error-free operation is achieved.
7. The error-free (syntax and semantic) and linked object code, in the form of a .HEX file, is transferred to an EPROM programmer. Fitted with a suitable adaptor, the EPROM programmer can "burn" the object file into the targeted EPROM-based 80C51-family device. For ROM-based devices, the object file is transferred to the factory for custom masking.

PROGRAM DESCRIPTION

Figure 7 is a flowchart of the program; following the flowchart is the program listing. The flowchart shows the basic processing and flow, while the listing documents the details of the program's implementation.

The program consists of four interrupt-driven (i.e., foreground) routines and a main program (i.e., background). The background program is entered at reset and executes forever, interrupted periodically by the foreground interrupts. Communication between the background program and the foreground handlers is via shared variables.

The four interrupt routines are as follows.

- multiplex () (INT3)

Free-running Timer I generates an interrupt at approximately 1000Hz and is used to multiplex the seven-segment and discrete LED display data. In a round-robin manner, at each interrupt, the program turns off the previously enabled display and writes data to, and enables, the next display. Finally, the interrupt routine sets a pointer to the next display—at the next interrupt, that display will be refreshed. Thus, each display (tens, ones, tenths, discrete LEDs) will be refreshed every fourth interrupt, which is more than fast enough for a flicker-free display.

Airflow measurement using the 83/87C752 and "C"

- read_switch () (INT6)

The PWM prescaler is configured to generate a periodic interrupt (INT6) at about 97Hz. The program counts these interrupts, and every 32nd interrupt sets an "update" variable. The main program will change the display data when it detects that "update" is set and clear "update" to prepare for the next display cycle. Thus, display change frequency is about 33Hz (i.e., 33ms), which eliminates display glitches associated with pushbutton switch bounce.

- calc_cfm () (INT0)

The air velocity turbine tachometer drives the 8XC752 INT0 interrupt pin. At each interrupt, the program reads Timer 0, which keeps track of the elapsed time (the low 16 bits of a 24-bit count in microseconds) between INT0 interrupts. The high-order 8-bit elapsed time count is cleared for possible updating by the following routine.

- overflow () (INT1)

When Timer 0 overflows (generating an interrupt), the program increments the high-order 8 bits of a 24-bit variable, counting the microseconds between tachometer interrupts (handled by the previous routine). If this 8-bit value becomes too large (i.e., tachometer interrupts stop), a NOFLOW variable is set, which will cause the main program to display an EEE out-of-range indicator on the seven-segment LEDs.

With the interrupt handlers executing the low-level timing and I/O, the main program, which is entered on reset and executes forever, consists of only three major steps.

The temperature/pressure compensated airflow is calculated. First, the "base" cfm rate, as tracked by the calc_cfm () tachometer interrupt is adjusted by removing the execution time of the calc_cfm () handler itself. Next, the temperature is determined (A/D channel 1), and airflow is compensated. Similarly, the air pressure is determined (A/D channel 0) and airflow compensated again.

Now that the true airflow is calculated, it is compared with the setpoint (adjusted with the variable resistor), which is determined by reading A/D channel 2. If the airflow is greater than the setpoint, the relay is closed. Otherwise, the relay is opened.

Finally, the UPDATE flag (set by the 33Hz read_switch () interrupt) is checked. If it is time to update, the data to be displayed is determined based on the pushbutton status and the state of the NOFLOW flag. The updated display data is initialized for later display on the LEDs by the multiplex () display refresh interrupt handler.

Airflow measurement using the 83/87C752 and "C"

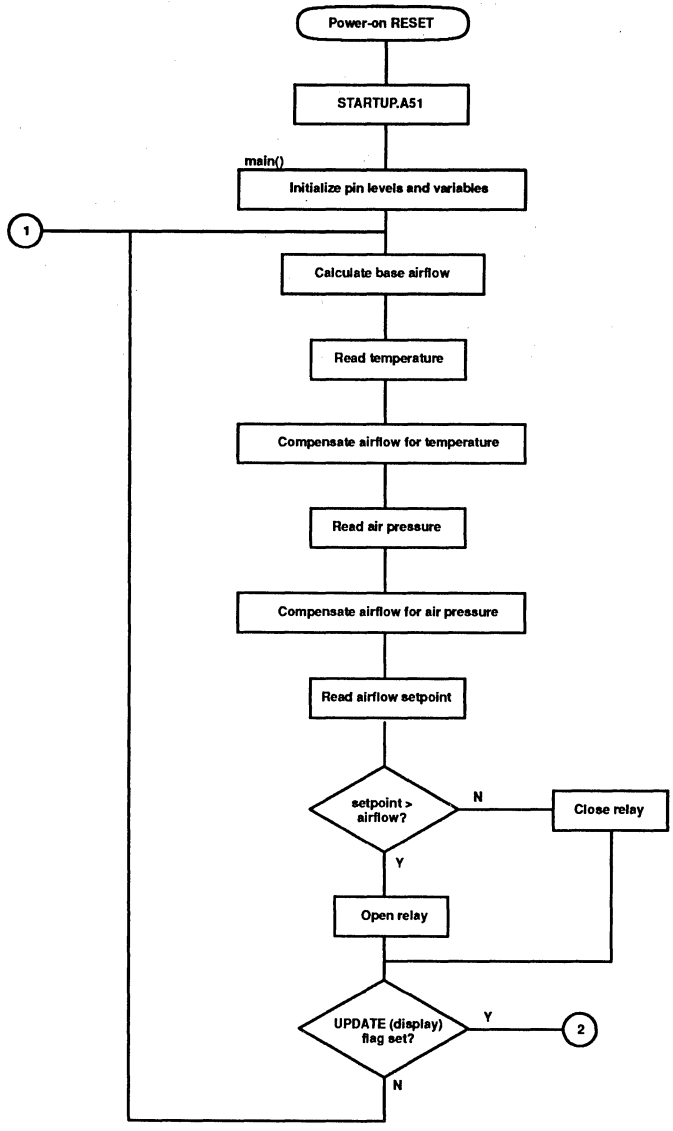


Figure 7. Program Flowchart

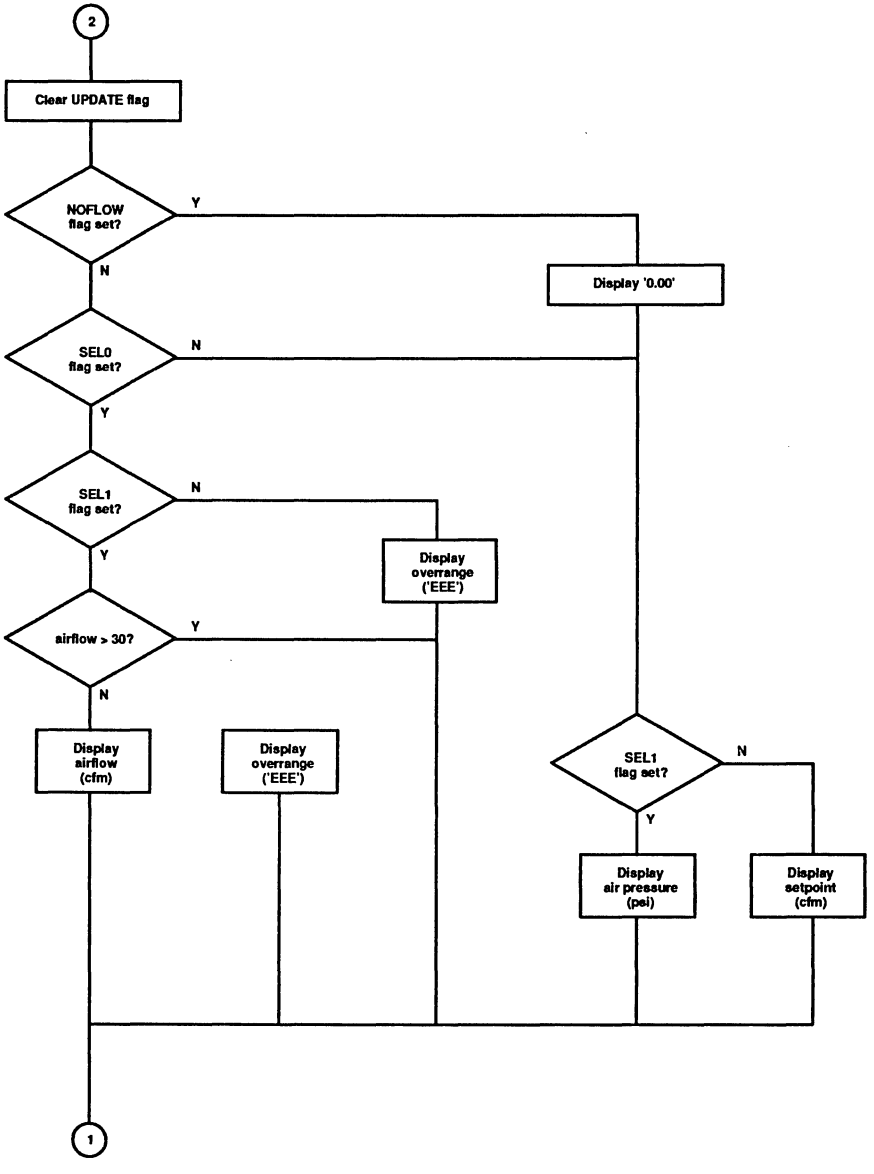


Figure 7. Program Flowchart (Continued)

Airflow measurement using the 83/87C752 and "C"

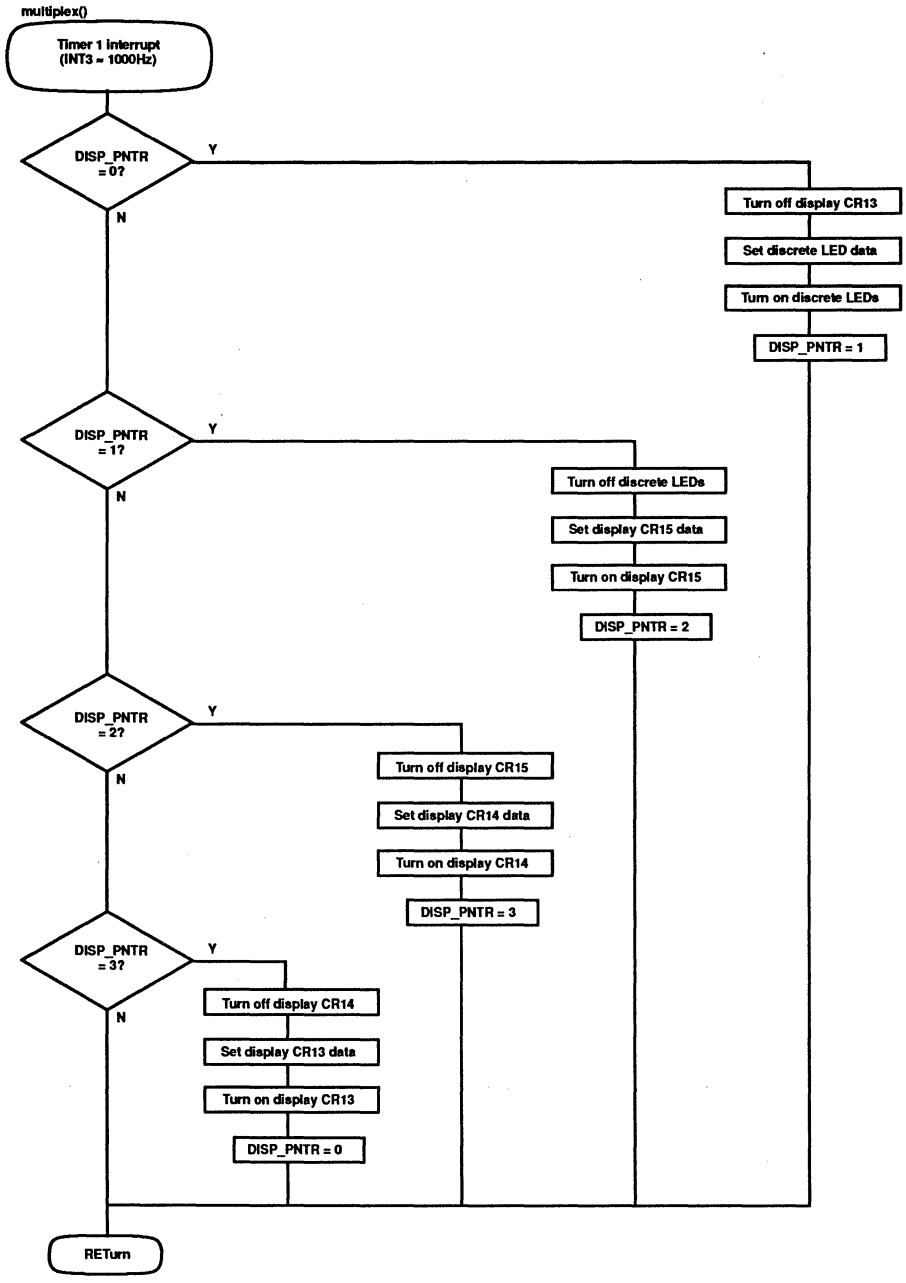


Figure 7. Program Flowchart (Continued)

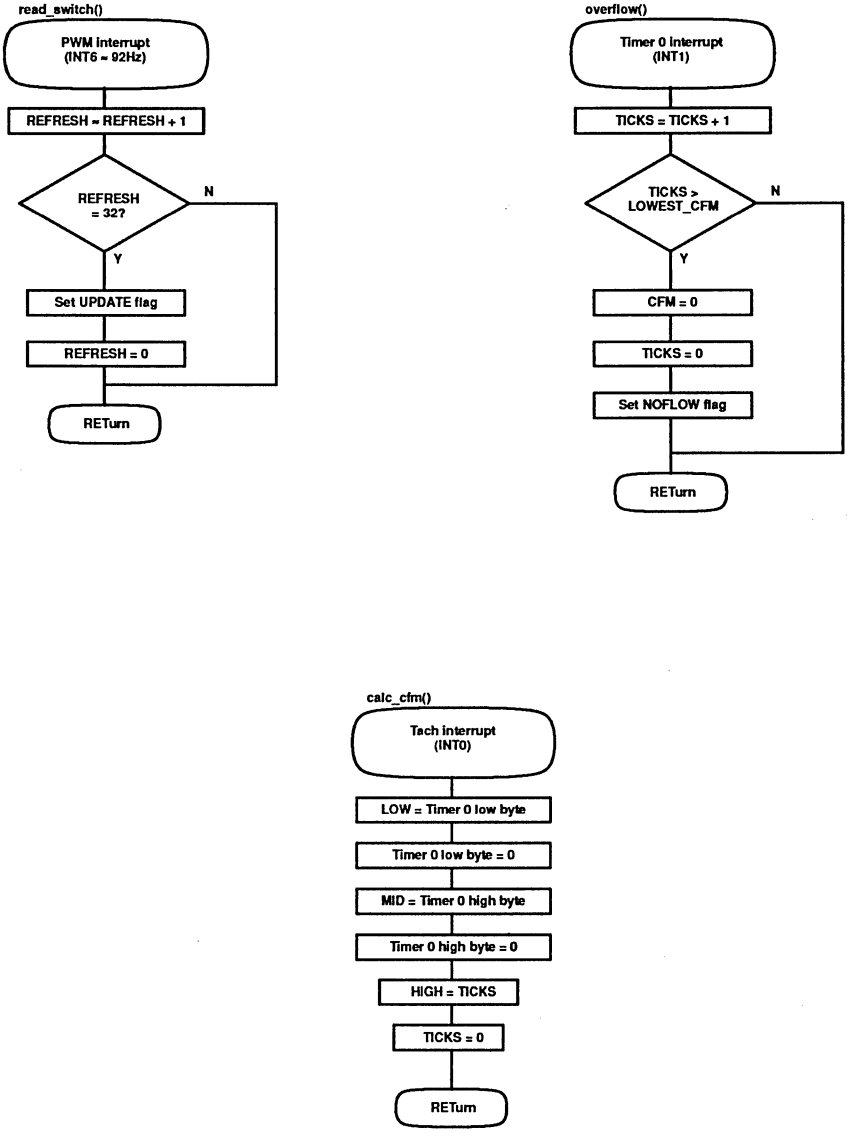


Figure 7. Program Flowchart (Continued)

Airflow measurement using the 83/87C752 and "C"

AN429

```
/*
                    this program measures the air flow through a rotary flowmeter
                    and displays the calculated cfm. the output of the flowmeter
                    tachometer is a small duty cycle pulse train with period
                    which is proportional to the flow. the flow is compensated
                    for changes in pressure and temperature to maintain
                    calibration. if the flow exceeds an adjustable setpoint
                    it energizes a 2 form c relay for user application.
*/

*/
                    these pragmas specify compiler command line options
*/
#pragma CODE                                /* generate code                                    */
#pragma SYMBOLS                             /* and symbols                                      */
#pragma PL (60)                             /* 60 lines per page                                */
#pragma PW (120)                            /* 120 cols per page                                */
#pragma OT (3)
#pragma ROM (SMALL)                         /* single-chip mode                                 */

*/
                    include the 8XC752-specific definitions and
                    the standard i/o library.
*/
#include                                      <reg752.h>
#include                                      <stdio.h>

*/
                    define symbolic names for program constants
*/
#define   ZERO_K              2730        /*  0 degrees centigrade in 1/10 kelvin             */
#define   ONE_TENTH_CFM       4444444L    /*  1/10 cfm in microseconds                        */
#define   STD_TEMP            2980        /*  25 degrees centigrade in 1/10 kelvin            */
#define   STD_ATM             147         /*  one atmosphere in 1/10 psi                      */
#define   LOWEST_CFM          0x40        /*  maximum period from meter 0x400000              */
#define   START_ADC0          0x28        /*  commands to start appropriate                   */
#define   START_ADC1          0x29        /*  a/d channel conversion cycle                    */
#define   START_ADC2          0x2a        /*                                                  */
#define   START_ADC3          0x2b        /*                                                  */
#define   START_ADC4          0x2c        /*                                                  */
#define   ADCI                0x10        /*  a/d converter status flags                      */
#define   ADCS                0x08        /*                                                  */
#define   FREERUN_I           0x10        /*                                                  */
#define   SEG_A               0x01        /*  P3 position for display segment 'a'             */
#define   CFM                 0x01        /*  P3 position for 'cfm' led                       */
#define   SEG_B               0x02        /*  P3 position for display segment 'b'             */
#define   DEGREES             0x02        /*  P3 position for 'degrees' led                   */
#define   SEG_C               0x04        /*  P3 position for display segment 'c'             */
#define   PSI                 0x04        /*  P3 position for 'psi' led                       */
#define   SEG_D               0x08        /*  P3 position for display segment 'd'             */
#define   SETPOINT            0x08        /*  P3 position for 'setpoint' led                  */
#define   SEG_E               0x10        /*  P3 position for display segment 'e'             */
#define   SEG_F               0x20        /*  P3 position for display segment 'f'             */
#define   SEG_G               0x40        /*  P3 position for display segment 'g'             */
#define   SEG_DP              0x80        /*  P3 position for display decimal pt.             */

typedef unsigned char byte;   /*          byte data type is unsigned 8-bit                    */
typedef unsigned int word;    /*          word data type is unsigned 16-bit                   */
typedef unsigned long 1_word; /*          1_word data type is unsigned 32-bit                 */
```

Airflow measurement using the 83/87C752 and "C"

AN429

```
#define TRUE 1                  /*   define logical true / false                    */
#define FALSE 0                 /*   values for bit variables                       */


/*
                define look-up table of possible seven segment display
                characters. the table consists of 11 elements corresponding
                to the 10 digits ('0'-'9') and error symbol ('E') that can be
                displayed. Each element is defined by ANDing (|) the bit
                mask for each segment (SEG_A - SEG_G) comprising the
                character. the table contents need to be inverted before
                use to be compatible with U2 (udn2585a). for example,
                '~segments[3]' specifies the segment mask to display '3'.

*/
code byte segments [ ] =
{
                SEG_A  | SEG_B  | SEG_C  | SEG_D  | SEG_E  | SEG_F           ,       /* 0 */
                         SEG_B  | SEG_C                                     ,       /* 1 */
                SEG_A  | SEG_B  |          SEG_D  | SEG_E  |                 ,       /* 2 */
                SEG_A  | SEG_B  | SEG_C  | SEG_D  |                  SEG_G  ,       /* 3 */
                         SEG_B  | SEG_C  |                  SEG_F  | SEG_G  ,       /* 4 */
                SEG_A  |          SEG_C  | SEG_D           | SEG_F  | SEG_G  ,       /* 5 */
                SEG_A  |          SEG_C  | SEG_D  | SEG_E  | SEG_F  | SEG_G  ,       /* 6 */
                SEG_A  | SEG_B  | SEG_C                                     ,       /* 7 */
                SEG_A  | SEG_B  | SEG_C  | SEG_D  | SEG_E  | SEG_F  | SEG_G  ,       /* 8 */
                SEG_A  | SEG_B  | SEG_C  | SEG_D  |          SEG_F  | SEG_G  ,       /* 9 */
                SEG_A  |                   SEG_D  | SEG_E  | SEG_F  | SEG_G          /* E */
} ;

/*              define the '752 special function bits which control i/o lines.
                note that i/o line (and constant) names are capitalized         */
sbit            RELAY       = 0x96;     /*   active hi to turn on setpoint relay   */
sbit            STROBE_0    = 0x80;     /*   active hi to enable display status led's */
sbit            STROBE_1    = 0x81;     /*   active hi to enable display cr15 (tenths) */
sbit            STROBE_2    = 0x82;     /*   active hi to enable display cr14 (ones) */
sbit            NO_FLOW     = 0x83;     /*   flag set when no flow detected        */
sbit            STROBE_3    = 0x84;     /*   active hi to enable display cr13 (tens) */
sbit            SEL_0       = 0x93;     /*   active low pushbutton inputs used to  */
sbit            SEL_1       = 0x94;     /*   select the display mode               */
sbit            INTR        = 0x95;     /*                                         */
sbit            UPDATE      = 0x97;     /*   flag set when time to update display  */

/*              define memory variables. note memory variable names are lower case */
data            word    cfm;            /*      gas flow in tenths of a cfm        */
data            word    setpoint;       /*      relay setpoint in tenths of a cfm  */
data            word    degree_c        /*      temperature in tenths centigrade   */
data            1_word  corr;           /*      intermediate calculation value     */
data            word    psi;            /*      pressure in tenths of a psi        */
data            byte    display0;       /*      variables to hold values for the   */
data            byte    display1;       /*      displays during refresh.           */
data            byte    display2;       /*      display0=status LEDs, display1=CR15, */
data            byte    display3;       /*      display2=CR14, display3=CR13       */
data            byte    disp_pntr;  /*   pointer to next display to enable       */
data            byte    refresh;        /*      counter determines display updates */
data            byte    high;           /*      bits 16 - 23 of flow period        */
data            byte    middle;         /*      bits 8 - 15 of flow period         */
data            byte    low;            /*      bits 0 - 7 of flow period          */
data            byte    ticks;          /*      incremented by timer overflow      */
```

Airflow measurement using the 83/87C752 and "C"

AN429

```
/*              the program consists of four interrupt handlers (multiplex,
                read_switch, overflow, calc_cfm) and a main program.
                 multiplex - refresh the seven-segment and discrete status LEDs
                 read_switch - signal periodic pushbutton sampling and display update
                 overflow - accumulate high order bits of time between tach pulses
                 calc_cfm - accumulate low order bits of time between tach pulses
                 main - calc airflow, control relay, sample pushbuttons, update display
*/

/*
                   multiplex -
                   use the free-running I timer to multiplex the seven-segment and
                   discrete leds at approx. 1000 hz.
*/

void multiplex () interrupt 3
{
                   switch(disp_pntr)
                   {
                          case 0x00:
                               STROBE_3 = FALSE;          /*  turn off display cr13   */
                               P3 = 0xff;                 /*  turn off all segments   */
                               P3 = display0;             /*  load segments for led's */
                               STROBE_0 = TRUE;           /*  turn on status led's    */
                               disp_pntr = 1;             /*  increment ptr to display                */
                               break;
                          case 0x01:
                               STROBE_0 = FALSE;          /*  turn off status led's   */
                               P3 = 0xff;                 /*  turn off all segments   */
                               P3 = display1;             /*  load segments for tenths                */
                               STROBE_1 = TRUE;           /*  turn on display cr15    */
                               disp_pntr = 2;             /*  increment ptr to display                */
                               break;
                          case 0x02:
                               STROBE_1 = FALSE;          /*  turn off display cr15   */
                               P3 = 0xff;                 /*  turn off all segments   */
                               P3 = display2;             /*  load segments for units */
                               STROBE_2 = TRUE;           /*  turn on display cr14    */
                               disp_pntr = 3;             /*  increment ptr to display                */
                               break;
                          case 0x03:
                               STROBE_2 = FALSE;          /*  turn off display cr14   */
                               P3 = 0xff;                 /*  turn off all segments   */
                               P3 = display3;             /*  load segments for tens  */
                               STROBE_3 = TRUE;           /*  turn on display cr13    */
                               disp_pntr = 0;             /*  increment ptr to display                */
                   }
}
```

```
/*
                    read_switch -
                    use the free running pwm prescaler to generate
                    interrupts at 92 hz. every 32nd interrupt set
                    the UPDATE flag which causes main () to sample
                    the pushbuttons and update the led displays.
*/

void read_switch () interrupt 6
{
                    if (refresh++ == 32)
                    {       UPDATE = TRUE;
                            refresh = 0;
                    }
}

/*
                    overflow -
                    whenever time0 overflows (from 0xffff to 0x0000)
                    increment the variable 'ticks' which accumulates the
                    highest order (16 - 23) bits of the gas flow period
                    in microseconds. if the variable 'ticks' is greater
                    than the period corresponding to a flow of < 0.1 cfm
                    then set the NO_FLOW flag which causes main () to
                    display '00.0'
*/

void overflow () interrupt 1
{
                    if (++ticks > LOWEST_CFM)
                    {
                            cfm = 0;
                            ticks = 0;
                            NO_FLOW = TRUE;
                    }
}

/*
                    calc_cfm -
                    an external interrupt (int0) generated by a tach
                    pulse from the flowmeter transfers the current value
                    of timer0 into variables 'low' and 'middle', and then
                    resets the timers. the 'ticks' variable described
                    above is also copied to variable 'high', and then
                    reset to zero. the NO_FLOW flag is cleared to
                    enable display by main () of the calculated cfm.
*/

void calc_cfm () interrupt 0
{
                    low = TL0;
                    TL0 = 0;
                    middle = TH0;
                    TH0 = 0;
                    high = ticks;
                    ticks = 0;
                    NO_FLOW = FALSE;
}
```

Airflow measurement using the 83/87C752 and "C"


```
/*
                main -
                after initializing pins and variables, enter a continuous loop to...
                - calculate the airflow based on the tach, temp and pressure inputs.
                - compare the airflow to the setpoint input, and control the relay.
                - if the UPDATE flag is set (by the read_switch interrupt handler),
                  sample the pushbuttons and update the display data.
*/

void main ()
{
                RELAY       = 0;            /*  initialize output pins                  */
                INTR        = 1;
                UPDATE      = 1;
                STROBE_0    = 0;
                STROBE_1    = 0;
                STROBE_2    = 0;
                STROBE_3    = 0;
                NO_FLOW     = 0;
                I2CFG       = FREERUN_I;    /*  enable I timer to run, no i2c           */
                RTL         = 0;            /*  timer 0 period 0x10000 u_seconds        */
                RTH         = 0;
                PWMP        = 255;          /*  pwm timer interrupt at 923 hz           */
                TR          = 1;            /*  enable timer 0                          */
                IT0         = 1;            /*  INT0 is edge active                     */
                ticks       = 0;            /*  initialize variables                    */
                cfm         = 0;
                low         = 0;
                middle      = 0;
                high        = 0;
                degree_c    = 250;          /*  25.0 tenths degrees c                   */
                psi         = 147;          /*  14.7 tenths psi                         */
                corr        = 0;
                refresh     = 0;
                disp_pntr   = 0;
                IE          = 0xab;         /*  enable interrupts                       */

*/
                main execution loop, executes forever.
*/

                while(1)
                {

*/
                calculate base cfm rate - first create long word representing
                flow rate period in microseconds. then subtract the time
                overhead in servicing the routine 'calc_cfm'. then divide the
                period into the period for 1/10 cfm, to get flow rate in 1/10
                cfm resolution.

*/

                        corr = high * 0x10000L;
                        corr += (middle * 0x100L);
                        corr += low;
                        corr -= CORRECTION;
                        corr = ONE_TENTH_CFM / corr;
```

```
/*
                read temperature - measure output from the LM35 sensor,
                scaled by the AMP-02. the scaling results in a range
                of 0 to 51.0 degrees centigrade, in 0.2 degree steps.

*/

                        ADCON = START_ADC1;
                        while (ADCON & ADCS) ;
                        degree_c = ADAT;
                        degree_c *= 2;

*/
                compensate cfm rate for temperature - convert temperature
                into degrees kelvin, then divide it into the measured flow
                rate multiplied by the calibration temperature of the flow-
                meter in degrees kelvin. (nominal 25 degrees centigrade)
*/

                        corr *= STD_TEMP;
                        corr /= (ZERO_K + degree_c);


*/
                read pressure - measure output of the KP100A pressure trans-
                ducer, scaled by the AMP_02. the scaling results in a range
                of 0 to 25.5 psi, in 1/10 psi steps.
*/

                        ADCON = START_ADC0;
                        while (ADCON & ADCS) ;
                        psi = ADAT;

*/
                compensate cfm rate for pressure - multiply measured pres-
                sure and the calculated flow rate, and then divide it by
                the standard atmospheric pressure at sea-level. (nominal
                14.7 psi)

                        corr *= psi;
                        corr /= STD_ATM;
                        cfm = corr;

*/
                read setpoint pot to obtain setpoint in the range of
                0 - 25.5 cfm in 1/10 cfm steps.
*/

                        ADCON = START_ADC2;
                        while (ADCON & ADCS) ;
                        setpoint = ADAT;

*/
```

```
                test if cfm rate greater or equal to the
                setpoint, and if so then energize relay
*/

                if (setpoint > cfm)
                     RELAY = 0;
                else
                     RELAY = 1;

*/
                test if UPDATE flag has been set, and if so reset flag.
*/

                if (UPDATE)
                {
                     UPDATE = 0;

*/
                then test is the NO_FLOW flag has been set. if so then
                display '00.0' cfm
*/

                        if (NO_FLOW)
                        {
                          display0 = ~CFM;
                          display1 = ~segments[0];
                          display2 = ~(segments[0]   |  SEG_DP);
                          display3 = ~segments[0];
                        }

*/
                if the NO_FLOW flag was not set then read the display
                select pushbuttons, and display the appropriate data.
*/

                        else if (SEL_0)
                        {
                               if (SEL_1)
                               {

*/
                if no pushbutton is depressed then the default display is
                the flow rate in cfm. if the flowrate is greater than
                or equal to 30 cfm then display the overrange message
                'EEE', otherwise display the flow in 'XX.X' format.
*/

                                        if (cfm <= 300)
                                        {
                                          display0 = ~CFM;
                                          display1 = ~segments[cfm % 10];
                                          cfm /= 10;
                                          display2 = !(segments[cfm % 10]);
                                          cfm /= 10;
                                          display3 = ~segments [cfm % 10];
                                        }
```

```
                                        else
                                        {
                                          display0 = ~CFM;
                                          display1 = ~segments[10];
                                          display2 = ~segments[10];
                                          display3 = ~segments[10];
                                        }
                                    }
*/
        if the temp pushbutton (SW1) is pressed then display the air temperature.
*/
                                    else
                                    {
                                      display0 =  ~DEGREES;
                                      display1 =  ~segments[degree_c % 10];
                                      degree_c /= 10;
                                      display2 =  ~(segments[degree_c % 10] | SEG_DP);
                                      degree_c /= 10
                                      display3 =  ~segments[degree_c % 10];
                                    }
                            }
                            else
                            {
*/
        if the psi pushbutton (SW2) is pressed then display the air pressure.
*/
                                    if(SEL_1)
                                    {
                                      display0 = ~PSI;
                                      display1 = ~segments[psi % 10];
                                      psi /= 10;
                                      display2 = ~(segments[psi % 10] | SEG_DP) ;
                                      psi /= 10;
                                      display3 = ~segments[psi % 10] ;
                                    }
*/
        if the setpoint pushbutton (SW3) is pressed then display the setpoint.
*/
                                    else
                                    {
                                      display0 =  ~SETPOINT;
                                      display1 =  ~segments[setpoint % 10] ;
                                      setpoint /= 10;
                                      display2 =  ~(segments[setpoint % 10] | SEG_DP ;
                                      setpoint /= 10;
                                      display3 =  ~segments[setpoint % 10] ;
                                    }
                                }
                            }
                        }
}
```

9XC1XX
Philips 16/32-bit microcontroller series

AN431

This Application Note is an updated version of an article published earlier this year and presented at Electro '90.

INTRODUCTION

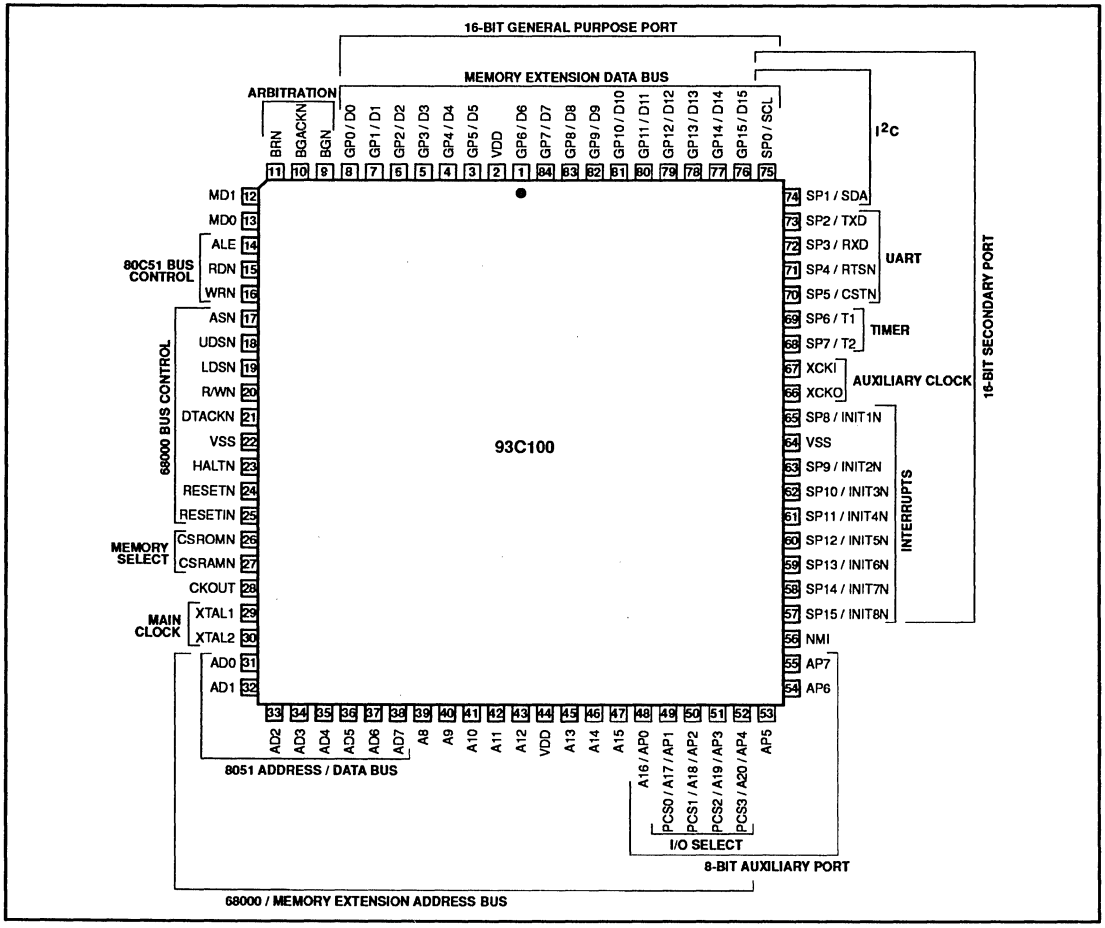
Philips 9XC1XX 16/32 bit microcontroller family introduces an innovative solution to 16-bit design by extending the 68000 design environment to 80C51 peripheral users. Based upon Philips 68070 microprocessor, the 9XC1XX line supports the full 68000 instruction set while providing up to 34K ROM, 512 bytes RAM, 256 Bytes EEPROM, UART, I²C bus port, two counter/timers plus 40 quasi-bi, and bidirectional I/O lines. The EPROM version will house 32K Bytes of memory programmable by standard 80C51 programmer with a 90C adaptor.

In addition to the on-chip peripherals the architecture has been updated to include decoded, latched external interrupts, an auto-DTACKN generator and control registers for user definition of system operation. Power consumption varies from 80mA during normal operation down to 10mA for Idle Mode and 8mA for Stand-by mode. All family members are available in 84-pin PLCC or 80-pin QFP in commercial and industrial temperature ranges.

NOTE: For the remainder of this article, 90C refers to all family members unless otherwise stated.

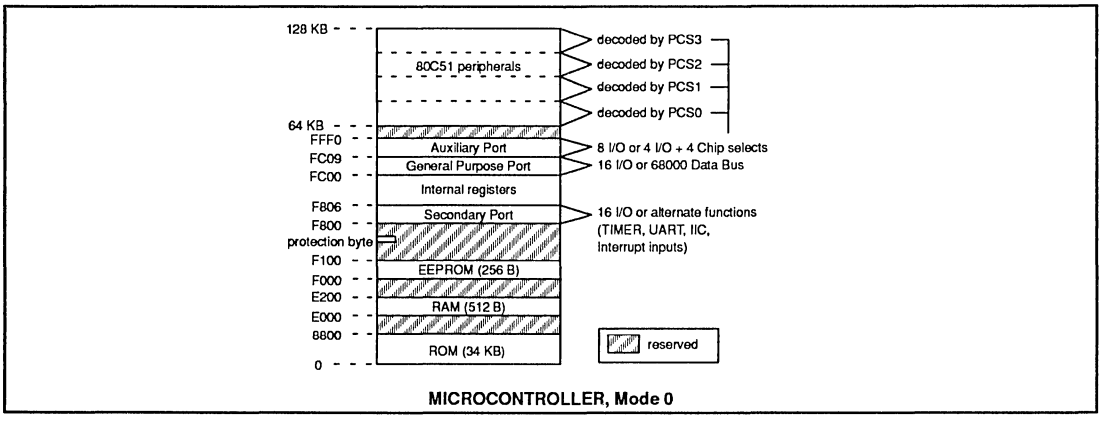
PINOUT

During RESET or power-up the 90C is configured to operate in one of four modes defined in hardware by the logic state of two input pins. The four modes differ primarily in external address range and I/O ports availability. Mode selection will depend upon the application's memory requirements vs. the need for I/O control. Once the operating mode has been established, port functions and peripherals are controlled by software access to on-chip memory locations.

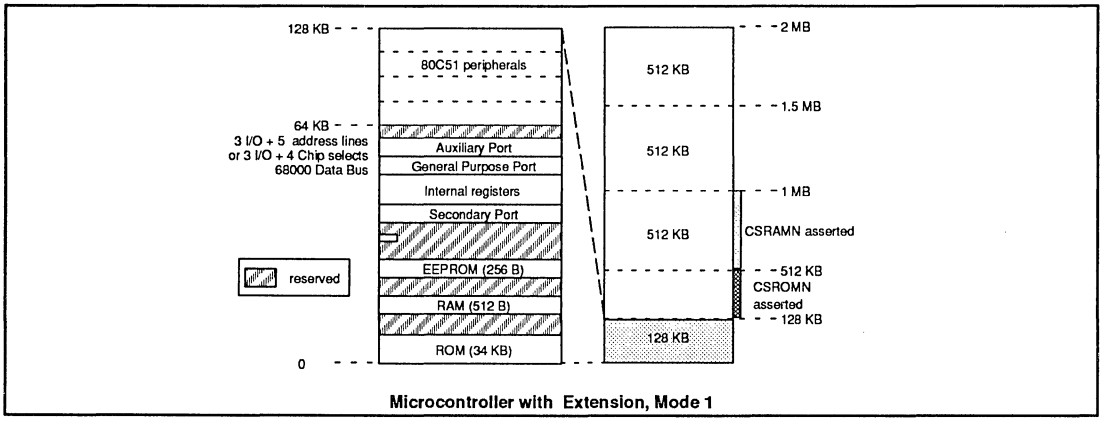


9XC1XX
Philips 16/32-bit microcontroller series

AN431



MICROCONTROLLER, Mode 0



Microcontroller with Extension, Mode 1

MODE 0

Microcontroller mode is used for embedded applications requiring only minimal memory (34K) space but extensive I/O and peripheral interface. The three ports, the General Purpose Port, Secondary Port and Auxiliary Port are available for I/O, peripheral functions and 80C51 bus chip selects. Each pin is individually definable as I/O or alternate function.

This provides a total of 40 I/O lines or 24 I/O lines, all on-chip peripherials plus 8 external interrupts or 20 I/O lines, an 80C51 interface, all of the peripherals and 4 interrupts. The designer not only has multiple combinations of features available but features which are defined by software allowing the system to be re-configured by service routines as the operating environment changes.

Mode 0 memory map divides the 128K memory into two 64K blocks reserving the lower 64K for on-chip functions. Access to the first 34K block are directed to the on-chip ROM. The next 30K maps peripheral operation, system control registers, RAM and EEPROM. The second 64K block defines the address range for the 80C51 bus. Access to locations $10000 to $1FFFF follow the synchronous 80C51 protocol supported by ALE, WRN and RDN signals. This block can be divided into four 16K blocks controlled by Auxiliary Port lines AP[1:4] enabled as chip selects PSEN [0:3]. The 64K 80C51 address block is the only available external address space. All other cycles are directed to on-chip locations.

MODE 1

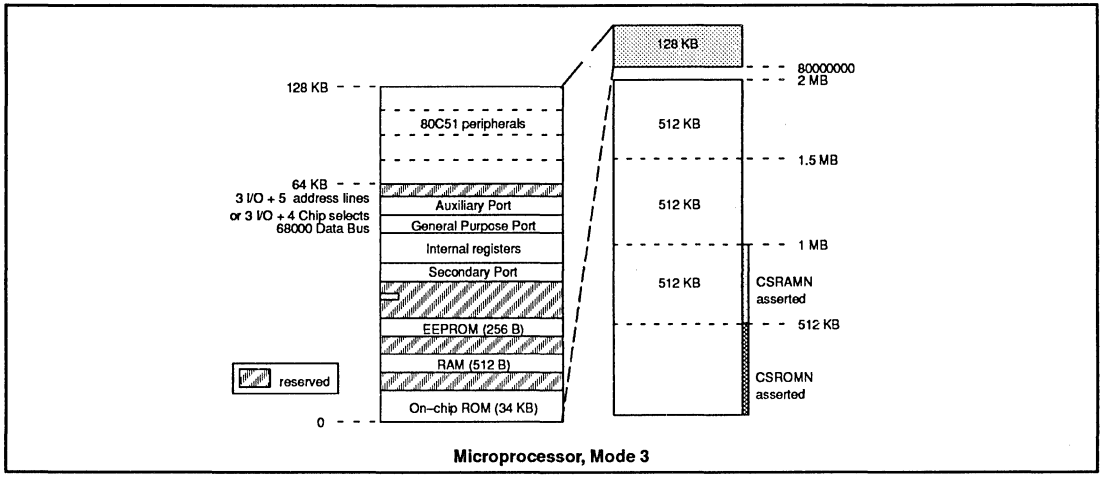
MODE 1 offers a flexible option for designs which require only 34K of space now but may need up to 2 Meg for future upgrades.

External access, beyond the first 128K address block, uses the eight, 80C51 A/D lines combined with either AP[1:4] to give 2 Mega bytes external range or AP[1] for a 128K extension. The remaining AP lines act as quasi-bidirectional pins or can be enabled to provide the 80C51 bank chip selects. Data for external transfers outside the first 128K is handled over the 16-bit general purpose port which provides a dedicated data bus. The secondary port is still fully operational to support any combination of I/O, interrupt input and peripheral functions.

Mode 1 provides the easiest upgrade path when moving from 8-bit 80C51 family designs to 16-bit applications. By taking advantage of the 80C51 bus, memory can be expanded up to 2 Mega bytes without significant changes to the peripheral structure. Firmware engineers can design using 68000 C-compilers in place of INTEL assemblers.

9XC1XX
Philips 16/32-bit microcontroller series

AN431



Microprocessor, Mode 3

MODE 2

Referred to as Emulation Mode, Mode 2 operation is identical to Mode1 with one major exception. Access to the first 34K of on-chip ROM is now decoded off-chip. This provides for memory emulation for code development, prior to final mask definition, without using an EPROM version. This Mode is particularly important to designers using the 93C110 part, which does not have an EPROM twin. However, duplication of the on-chip timing does require either DTACKN to be generated externally or 266ns EPROMs used to mimic the 4 clock cycle internal transfer time. The remainder of the memory map is unaffected and models MODE 1.

MODE 3

This last mode creates a memory map that is almost an inversion of the Mode1 map. All on-chip memory locations reside in the upper 4G bytes of the 32-bit internal address bus. On-chip ROM, RAM, registers, etc., begin at $80000000 with the lower 2 Meg of memory filled by off-chip locations $0000 to $3FFFFF. In this mode the 90C behaves as a microprocessor drawing instructions from board memory. With the exception of new address locations, on-chip operation is as with Mode 1.

Now that you understand how the 90C operates, the remainder of the article will concentrate on unique features and how the part compares to the 68000 and the 68070.

THE CPU

The Central Processing Unit for the 90C family maintains the 68000 architecture programming model, instruction set and addressing modes. Any 68000 compiler or assembler can be used for code translation as long as the programmer considers the differences in information stored during exception processing. When the 68000 enters exception processing, the CPU stacks three words; program counter high, program counter low and the status word. The 90C stores one additional word, containing format bits and the vector number, when responding to normal exceptions. 13 additional words are stored for address exceptions to allow the CPU to recover and re-run the lost cycle.

Interrupts

The 90C has 29 potential interrupt sources individually programmable to seven priority levels:

GP[0:15]	A change in status of an input at the GP port.
INT[1:8]N	External Latched Interrupts
UART	Receiver and Transmitter
I^2C	Status
T1, T2	Status
NMIN	External Non-maskable interrupt, level 7.

The problem of simultaneous requests becomes a more critical concern as the number of interrupt sources increases. This situation is resolved in two ways. First, to prevent stack overflow, a bit set to the System Control Register will delay service to subsequent higher level requests once an interrupt routine has started. Pending interrupts are recognized and serviced according to priority when the bit is cleared at prior to RTE of the current interrupt. This feature allows greater software control over interrupt prioritization while significantly reducing the amount of dedicated stack space required to implement a multiple interrupt source design.

The second improvement adds an internal prioritization scheme to handle simultaneous interrupts from multiple sources programmed to the same priority level. The scheme is similar to the 68070 prioritization list but includes the NMIN and external latched requests giving them highest priority. For example, simultaneous interrupts from INT2N and the UART receiver, both programmed to level three are acknowledged by completing the INT2N requests first followed by the UART receiver request.

DTACKN Generator

Designers using external memory or peripherals mapped outside of the 80C51 address range (i.e., Modes 1–3) are ultimately faced with the need for a DTACKN generator, preferably one which optimizes rather than degrades system performance. This hardware overhead is eliminated by

taking advantage of the ADD and FBC bits in the System Control Register to optimize READ cycles and enable auto-DTACKN control. When the Fast Bus Cycle bit (FBC) is set and DTACKN is pulled low, the minimum number of external READ bus cycles is reduced from four to three. With FBC asserted, software transferring block data from external memory executes about 10% faster than normal.

If DTACKN cannot be pulled low, normally the 90C will insert wait states to delay the cycle until the acknowledge is received. However, an internal acknowledge is automatically generated after 7 clock periods when the ADD bit (Auto DTACKN Disable) is cleared. The maximum access time required is 250ns at 17.5MHz CPU.

Power Down Modes

Two bits in the System Control Register control the main and auxiliary oscillators to enable normal, idle or stand-by operating mode. As expected, during normal operation all chip functions are supported and the main oscillator is running at full speed. In IDLE mode, power consumption is reduced by driving the chip from the auxiliary clock at 300KHz while the main oscillator remains running. This maintains minimal functionality while reducing power requirements to 50mW. Normal mode can be re-entered by external or internal interrupt. STAND-BY mode also drives the system from the auxiliary clock but the main oscillator is turn off to reduce power consumption to 40mW. The same functionality as IDLE mode is maintained, however, to return to normal operation the main oscillator must be restarted and stabilized for at least 10ms. During IDLE and STAND-BY modes some on-chip functionality is maintained (i.e., RAM integrity, timer and UART operation) and a minimal instruction set supported.

INTER-INTEGRATED CIRCUIT (I^2C) BUS

The I^2C port on the 90C family is a two-wire serial bus designed to support information transfers between multiple elements within a system rack or desk top range. The port operates as a master or slave transmitter or receiver and supports multimaster operation.

When operating as a master the SCL bus clock drives or receives byte data at transfer rates up to 189KHz (I^2C specification compatibility = 100KHz max). The port is controlled by access to memory mapped registers and operates in either polled or interrupt driven designs.

UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER

The UART interface follows the standard 2681/2692 programming models and operating modes. 8- or 7-bit characters are transferred at up to 19.2K baud using either a two- or four-wire handshake. Independent receiver and transmitter clocks are selected either from an onboard baud rate generator or an external source via the auxiliary clock. UART is initiated and controlled through a dedicated memory mapped register set.

16-BIT COUNTER TIMERS

An on-chip 16-bit continuous timer increments once every 192 CPU clocks to provide a reference for two 16-bit programmable timers operating in match, count, or event mode. Match mode generates a pulse output on the corresponding timer special function pin (T1 or T2). The pulse duty cycle equals Tx/($FFFF-RR) with duration $FFFF-RR where RR is a value programmed into the reload register and Tx is the T1 (T2) reference register. At rollover and match conditions an interrupt is issued and a status bit is set. In event counter mode, when a programmed number of events occurs at the T1 (T2) input, an interrupt is issued. An event is definable as +edge, –edge, ±edge or a level change. In capture mode, each time an event occurs, the content of the continuous timer is stored in the T1 (T2) register, an interrupt is issued and a status bit is set. Event mode is used to count the number of events occurring during a fixed time span while capture mode determines the time span between two events.

DEVELOPMENT TOOLS

When introducing a new microcontroller the second question customers always ask (after "What's new about it?") is "What development tools exist?". The 90C family is supported by three levels of evaluation hardware and high level software all of which are currently available either from a third party or via distribution.

The simplest and least expensive hardware support is the Microcore III evaluation/demo board from Philips. This is a spin-off from the 68070 Microcore I and offers the same features minus the VSC and video interface. The board consists of 4×6" card with sockets for 512K EPROM and RAM, an RS232 port, I^2C port and 93C100. There is also a prototype area and 96 pin connector. No memory is included but the on-chip ROM holds a 32K monitor which supports line assembly, uploads and downloads from a host, breakpoints, single step operation and a few other general purpose functions. The board operates by RS-232 serial connection to a PC running in terminal mode. The MCIII costs less than $600 and is great for initial software development and debugging assembly level driver routines.

The second level of support is the 90CDS, a full emulation system designed by Philips and sourced and supported by third party for the U.S. market. The system consists of a stand-alone box with I.C.E. extension driven by serial connection to a PC. Although a low-level monitor is supplied with the system, most detailed hardware analysis will require the 90C-XRAY development software package from BSO/TASKING, Boston. The package includes a C-compiler, Assembler/Linker/Librarian and Microtec XRAY debugger. XRAY supports C code and Assembly level symbolic debug, real time operation, hardware and software breakpoints and 2K of Trace memory. The user interface is a windowed environment with on-screen help. The 90CDS retails for under $6000 and the Tasking package sells for less than $4000.

The most sophisticated development system, the Lauterbauch TRACE 32, is available from SIGNUM systems in Thousand Oaks, CA. While this is the most expensive solution, it is also the most powerful. Like the HP 64700, the TRACE 32 is actually a host system which supports numerous 8-, 16- and 32-bit controllers through plug-in modules. The unit recognizes multiple high level languages and includes a built in logic analyzer. In addition to the 90C family, the TRACE 32 also supports most of Philips 80C51family and the 68070. For designs using more than one controller or a mixture of controller/processor architectures, this system provides a common data base and may be the most economical solution.

During the last 6 months the 16-bit processor/controller market has experienced a host of new product introductions from some healthy competition. However, in a side-by-side comparison, the 90C100 family wins out easily by providing more on-chip memory, standard instructions and more useable features than any other product in its class. The best news is that even with all these advantages it's still one of the lowest cost 16-bit products in the world.

Parallel port on the SBE 68070

AN432

INTRODUCTION

The SBE 68070 has a parallel port input to provide a fast way of downloading program. But unfortunately, this port is not supported by the on-board monitor.

So for those of you who can't afford to wait during the serial download, here is an implementation of a parallel link between a PC and the SBE.

This implementation consists of two programs:

- a C program on the host PC : SP (send parallel)
- a C program on the SBE : PR (parallel receive)

The program on the PC uses the parallel port LPT1 (or PRN) and can be easily modified to support another parallel port.

The program on the SBE can be implemented in ROM or can be downloaded as needed.

The following files are available on the Signetics BBS:

- SP.C C source code for SP
- SP.EXE Executable file for the PC
- PR.C C source file for the SBE
- PR.SRC assembler file generated by the compiler
- ENTRY.SRC entry level routine for C program
- PR.CMD command file for the linker
- PR.ABS S code file for PR.C

The BBS phone number is: (800)451-6644 or (408)991-2406.

These files are grouped under the name: SBEPAR.ARC

The S-code is starting at $F88000 and can be merged in the EPROMs of the SBE with Micromon.

FILE TRANSFER

The file to transfer should be a S-code file with no symbolic information in it, i.e., it should start by an S0 record. This file is transferred to the SBE via the parallel port and the decoding is done by the SBE.

This program recognizes all S-records types:

S0 : header
S1 : data record with 16 bits address
S2 : data record with 24 bits address
S3 : data record with 32 bits address
S4,S5,S6 : ignored
S7,S8,S9 : last record of the file

The checksum at the end of each record is ignored.

COMMANDS

To download a program into the SBE :

- Use a terminal emulator on your PC to connect to the SBE and start the PR program by issuing the command GO F88000.
- Go to a dos-shell and start the SP program. You can supply the name of the absolute file on the command line or the program will ask for it. During the download, SP displays one dot for each S-code record transferred.

At the end, both programs stop, SP returns to DOS prompt and PR ends with a breakpoint.

USING THIS PROGRAM WITH XRAY

You cannot use directly this program when XRAY is running. However, you can download your program before starting XRAY and when you load your program into XRAY, you can specify the /NI option which tells XRAY to load all the symbolic information but not to download the code to the SBE.

Remark: you need to link your program twice: one to generate IEEE format for XRAY, the other to generate S-code for the download.

HINTS AND TIPS

- Some controls lines used for the parallel port are connected to the 68681 Duart. The 68681.h file contains a full definition of the DUART register in C. A function which needs these definitions has to define the Duart variable and the DUART_Base symbol. This is done by inserting "USE_DUART" at the beginning of each function and by defining DUART_Base in the file. This variable definition allows the C compiler to generate the addressing mode $xx(an) which is more efficient to access peripherals.
- The Parallel port of the PC is not used as a printer port, the control lines are driven directly by the program. This program does not interfere with the XRAY hardware key

Here is a description of how the transfer works: The PC controls the lines SelectN and StrobeN, the SBE answers with AckN.

1. The PC makes sure that the parallel interface on the SBE is free (AckN high).
2. The PC sets the data on the data lines and asserts Strobe and SelectN, StrobeN is used by the hardware of the SBE to latch the data.
3. When the SBE has taken the data, it asserts AckN.
4. The PC releases StrobeN and SelectN and finally, the SBE releases AckN when it is ready for the next data.

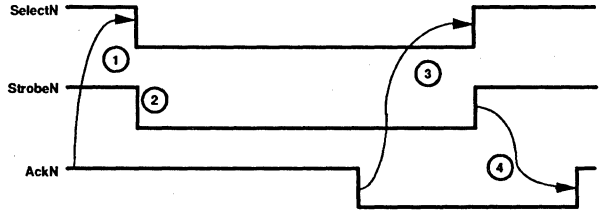


Figure 1. Control Lines Timing

Parallel port on the SBE 68070 AN432

– The Parallel port should be disconnected from the PC when the SBE is powered up: some voltage levels present on the cable may interfere with the reset sequence.

Rule of thumb: wait until the Halt LED of the SBE is switched off before connecting the parallel cable.

– The cable between the PC and the SBE is a wire to wire connection. If you are using the SBE 68070 version 1, it is not possible to plug the connectors for the serial line and the parallel port at the same time. The solution is to use a 2x50 pin connector for flat cable and connect both cables on this connector. (See Figure 2.)

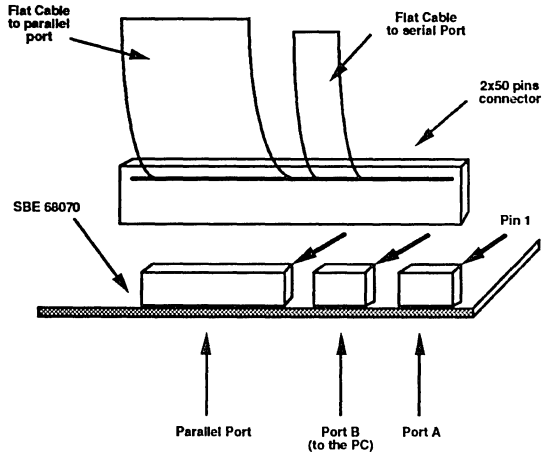


Figure 2.

LISTING OF THE PR PROGRAM

File : 68681.h

```
/*                DUART bit status register assigment:          */
/*                     SRA / SRB                                */

#define RXRDY                   0x01
#define FFULL                   0x02
#define TXRDY                   0x04
#define TXEMT                   0x04
#define E_OVERRUN               0x10
#define E_PARITY                0x20
#define E_FRAMING               0x40
#define BREAK                   0x80

/*
*                DUART registers:
*/

#define USE_DUART  register char * Duart = DUART_Base;

#define du_mra     *(Duart+0x1)  /* mode register              R/W */
#define du_sra     *(Duart+0x3)  /* status register            RO  */
#define du_csra    *(Duart+0x3)  /* clock select register      WO  */
#define du_cra     *(Duart+0x5)  /* command register           WO  */
#define du_rhra    *(Duart+0x7)  /* receiver holding register  RO  */
#define du_thra    *(Duart+0x7)  /* transmit. holding register WO  */

#define du_mrb     *(Duart+0x11) /* mode register              R/W */
#define du_srb     *(Duart+0x13) /* status register            RO  */
#define du_csrb    *(Duart+0x13) /* clock select register      WO  */
#define du_crb     *(Duart+0x15) /* command register           WO  */
#define du_rhrb    *(Duart+0x17) /* receiver holding register  RO  */
#define du_thrb    *(Duart+0x17) /* transmit. holding register WO  */

#define du_acr     *(Duart+0x9)  /* auxiliary control register WO  */
#define du_ipcr    *(Duart+0x9)  /* input port change register RO  */
#define du_isr     *(Duart+0xb)  /* interrupt status register  RO  */
#define du_imr     *(Duart+0xb)  /* interrupt mask register    WO  */
#define du_ivr     *(Duart+0x19) /* interrupt vector register  RW  */
#define du_ipr     *(Duart+0x1b) /* input port register        RO  */
#define du_opcr    *(Duart+0x1b) /* output port conf. register WO  */
#define du_op_set  *(Duart+0x1d) /* output port Set register   WO  */
#define du_start   *(Duart+0x1d) /* start counter register     RO  */
#define du_op_res  *(Duart+0x1f) /* output port Reset register WO  */
#define du_stop    *(Duart+0x1f) /* stop counter register      RO  */
```

Parallel port on the SBE 68070 AN432

File : PR.C

```
/*
    This program is intended to be used with the SBE 68070  It allows
the transfer of S-code through the parallel port of the SBE

*/

#include "68681.h"

#define UCHAR unsigned char
#define ULONG unsigned long

#define DUART_Base      (char *)0xFFFC00
#define Parallel        *(char *)0xFFFF85

#define Set_busy        du_op_res = 0x04
#define Set_ack         du_op_res = 0x80
#define Reset_busy      du_op_set = 0x04
#define Reset_ack       du_op_set = 0x80
#define Set_errorn      du_op_res = 0x08

UCHAR p_get()
{
char data;
USE_DUART

do {} while ( (du_ipcr & 0x08) != 0 ); /* wait for select low */
data = Parallel;
Reset_ack;
do {} while ( (du_ipcr & 0x08) == 0 ); /* wait for select high */
Set_ack;
return data;
}

UCHAR atoi(x)
UCHAR x;
{
if ( (x>='0') && (x<='9') ) return x-'0';
if ( (x>='A') && (x<='F') ) return x-'A'+10;
if ( (x>='a') && (x<='f') ) return x-'a'+10;
return -1;
}


UCHAR get_byte()
{
UCHAR data;
data = p_get();
data = atoi(data) << 4;
data |= atoi( p_get());

return data;
}

main()
{
UCHAR *address;
UCHAR data;
ULONG Addr;
int Bytecount,dl_end;
USE_DUART

dl_end = 0;
```

```
Set_busy;
Set_ack;
data = Parallel;

do
  do {} while (p_get() != 'S');      /* wait for the first S0 record */
while (p_get() != '0');

do {} while (p_get() != 0x0a);       /* wait for end of line  */


do
  {
      /* now process data records */

   do {} while (p_get() != 'S'); /* first character should be S */

   data = p_get();
   Bytecount = get_byte();

   switch (data)
      {case '1' :
        { Addr = get_byte() << 8;
          Addr += get_byte();
          address = (UCHAR *)Addr;
          Bytecount -= 3;
          for (; Bytecount > 0; Bytecount-- ) *address ++ = get_byte();
          break;
        }
       case '2' :
        { Addr = get_byte() ;
          Addr = (Addr << 8) + get_byte();
          Addr = (Addr << 8) + get_byte();
          address = (UCHAR *)Addr;
          Bytecount -= 4;
          for (; Bytecount > 0; Bytecount-- ) *address ++ = get_byte();
          break;
        }
       case '3' :
        { Addr  = get_byte();
          Addr  = (Addr << 8) + get_byte();
          Addr  = (Addr << 8) + get_byte();
          Addr  = (Addr << 8) + get_byte();
          address = (UCHAR *)Addr;
          Bytecount -= 5;
          for (; Bytecount > 0; Bytecount-- ) *address ++ = get_byte();
          break;
        }
       case '4' : ; case '5' : ; case '6':
        {
         do {} while (p_get() != 0x0a);   /* wait for end of line */
         break;
        }
       case '7' : ; case '8' : ;case '9' :
        {
        do {} while (p_get() != 0x0a);    /* wait for end of line */
        }                                 /* No break to go to default */
       default :
        { dl_end = -1;
        }
       }
    } while (dl_end==0);
}
```

Program SP

```
/*
 This program is intented to be used as a download program
 to the SBE 68070 V2. It uses the parallel port LPT1 to transmit
 the data.
*/

#include <conio.h>
#include <stdio.h>
#include <process.h>

#define UCHAR unsigned char

#define Data_latch      0x378                /* LPT1 port addresses */
#define Ctrl_out        0x37a
#define Ctrl_in         0x379

void p_put(UCHAR x)
{
do {} while ( ( inp(Ctrl_in) & 0x40 ) == 0 ); /* make sure ACKN is high /
outp(Data_latch,x);
outp(Ctrl_out,0x09);            /* Start Strobe pulse & Selectn pulse */
do {} while ( ( inp(Ctrl_in) & 0x40 ) != 0 );          /* Wait for ACKN /
outp(Ctrl_out,0x00);                    /* Strobe & Selectn go high */
}

main( int argc, char *argv[], char *envp[] )
{

FILE *fichier = NULL;
char answer[64];
int i;

outp(Ctrl_out,0x00);

if (argc == 2)
  {
  fichier = fopen (argv[1],"rt");
  if (fichier == NULL)
    {
     printf("Unable to find file");
     exit(2);
    }
  }
else
  {do
     {
      printf(" File to download to the SBE -->");
      i = scanf("%s",&answer);
      printf("\r\n");
      if (i==1) fichier = fopen (answer,"rt");
     }
   while (fichier == NULL);
  };

while ( (i = fgetc(fichier)) != EOF )
  {
   p_put((UCHAR)i);
   if (i == 'S') putchar('.');
  };

fclose(fichier);
}
```

"Opti-Mizer" power management for notebook computers using the 8XC752 microcontroller

AN436

INTRODUCTION

With conventional microprocessor based systems, the market was primarily concerned with performance, cost and features. With the advent of hand-held and portable computers, the prominent market requirements focus on size, weight and battery life.

Given a mature 386SX/AT architecture that provides more than adequate performance for average notebook application usage, the design challenges for these machines revolve around developing low power systems that maximize battery usage.

The features of a notebook PC are usually characterized as weight and battery life. The heaviest component of a PC is usually the battery and the choice of battery is dictated by the power required. Thus the performance of the power management scheme has a direct bearing on both these parameters.

The battery life targets for notebook machines is around 5 hours (the air commute time from coast to coast USA).

OVERVIEW

Most of the power consumed by a fully powered PC is wasted. The hard disk spins constantly even though data transfers from the disk are very sporadic. PCs may sit unattended for periods where the user is distracted by a telephone call, for instance.

There are two principle challenges in designing a power management system: the ability to power down various devices without affecting other devices on the same bus, and ensuring full compatibility with existing operating systems and applications.

The largest user of power in a PC is the display sub-system (3–5 Watts) followed by the peripherals such as the hard disk (2–4 Watts), the main system memory (0.5–1.5 Watts), and the core logic (1 Watt).

INTEGRATED POWER MANAGEMENT

The conventional PC architecture needs to be extended to support power management. Hardware needs to be added to provide power-down capabilities and software needs to be added to support the hardware and provide DOS compatibility. The software support is usually realized in the BIOS. The hardware support can be implemented with external circuitry. This external circuitry manages the power resources to individual sub-sections of the PC system as these resources dictate. The external circuitry monitors battery power, system activities and timed events.

Conventionally, the external circuitry is comprised of a digital power management ASIC and associated components. The use of this part increases the chip count of the PC system.

The power management system has to determine how the system resources are being used. The resource usage of a PC can be determined by monitoring events or activities. User activity is usually determined by monitoring the keyboard controller for keystroke events. Keystroke events can be indicated by interrupts to the PC core logic or IO reads to the keyboard controller location.

The power management ASIC solutions on the market today, such as the VADEM/INTEL 82C347, VLSI VL82C312 and INTEL 80C386SL all require external analog support circuitry to completely implement the power management functions. For example, low battery detect is implemented by the use of external comparator chains and complex, close tolerance level detect circuitry. The cost of this external circuitry is usually a significant proportion of the overall cost of the power management solution. The 83/87C752 employs an internal analog-to-digital converter (ADC). The ADC can be used to implement the battery level detection function at no extra cost and with no extra support circuitry.

The 83/87C752 is a member of the Philips/Signetics 8051 family of high performance 8-bit microcontrollers. These processors have been optimized for sequential real time control applications. The 83/87C752 contains most of the features of the 80C51 and has the following features:

- 2K bytes ROM
- 64 bytes RAM
- Single level interrupt structure
- 16 bit programmable counter/timer
- Two 8-bit and one 5-bit bi-directional IO ports
- I^2C serial interface
- PWM with interrupt and overflow capability
- 5 channels of 8-bit A/D
- 28-pin DIP and PLCC.

FLEXIBILITY

ASIC solutions to power management offer rigid schemes which work adequately with a few notebook architectures, but rarely offer exactly what the designer requires. With the current competitive arena for laptop development, time-to-market and value added features have a significant impact on the sales success of a particular product. The 83/87C752 offers flexibility at a low price. the power management design requirements can be coded and configured in the controller software and One Time Programmable (OTP) devices can offer a quick low-cost implementation of the coded scheme.

With these integrated functions that the 83/87C752 offers, and its ability to provide a complete solution to power resource control, this device is emerging to be the industry standard for power management.

TOPOLOGY

Figure 1 shows a block diagram of a typical system implementation. It employs the integrated power management scheme using the Philips/Signetics microcontroller to handle the keyboard and power management functions. The CPU and coprocessor reside on the local bus with the system memory. A local bus controller monitors CPU bus cycles to see if they are memory or ISA cycles. It also integrates the interrupt and DMA functions. The ISA bus controller processes non-system memory bus cycles. A frequency generator is used to provide the system clocks and clock multiplexing. The peripheral controller integrates the communications and mass storage sub-system. The VGA sub-system shares the ISA bus with the peripheral controller. The VGA controller has associated VGA memory.

Figure 2 shows the microcontroller with the external support devices. The frequency generator provides the system clocks. It must have the ability to change the frequency of the clocks without violating the minimum high or low times for the core logic. The Philips/Signetics microcontroller can control the speed of the system clocks via frequency select pins on the frequency generator. Frequency generators such as the Avasem AV9127 can change the processor clock speed gradually and continuously without violating the minimum high or low times.

The integrated controller monitors system activity via its digital input ports. It uses internal timers to time the intervals between activity. The power to the VGA and peripheral sub-systems is controlled by the digital output port pins via MOSFETs.

Battery level and V_{CC} is monitored by the onboard A/D converter.

"Opti-Mizer" power management for notebook computers using the 8XC752 microcontroller

AN436

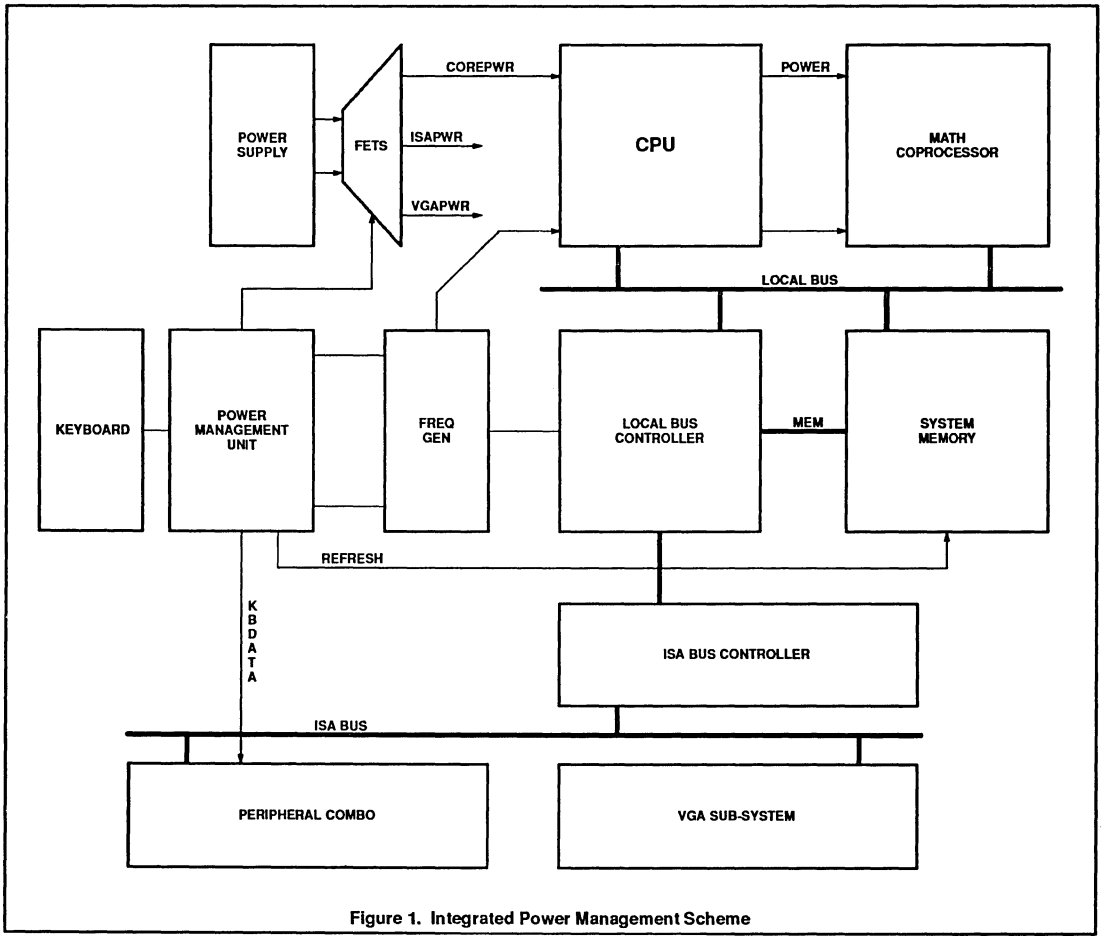


Figure 1. Integrated Power Management Scheme

"Opti-Mizer" power management for notebook computers using the 8XC752 microcontroller

AN436

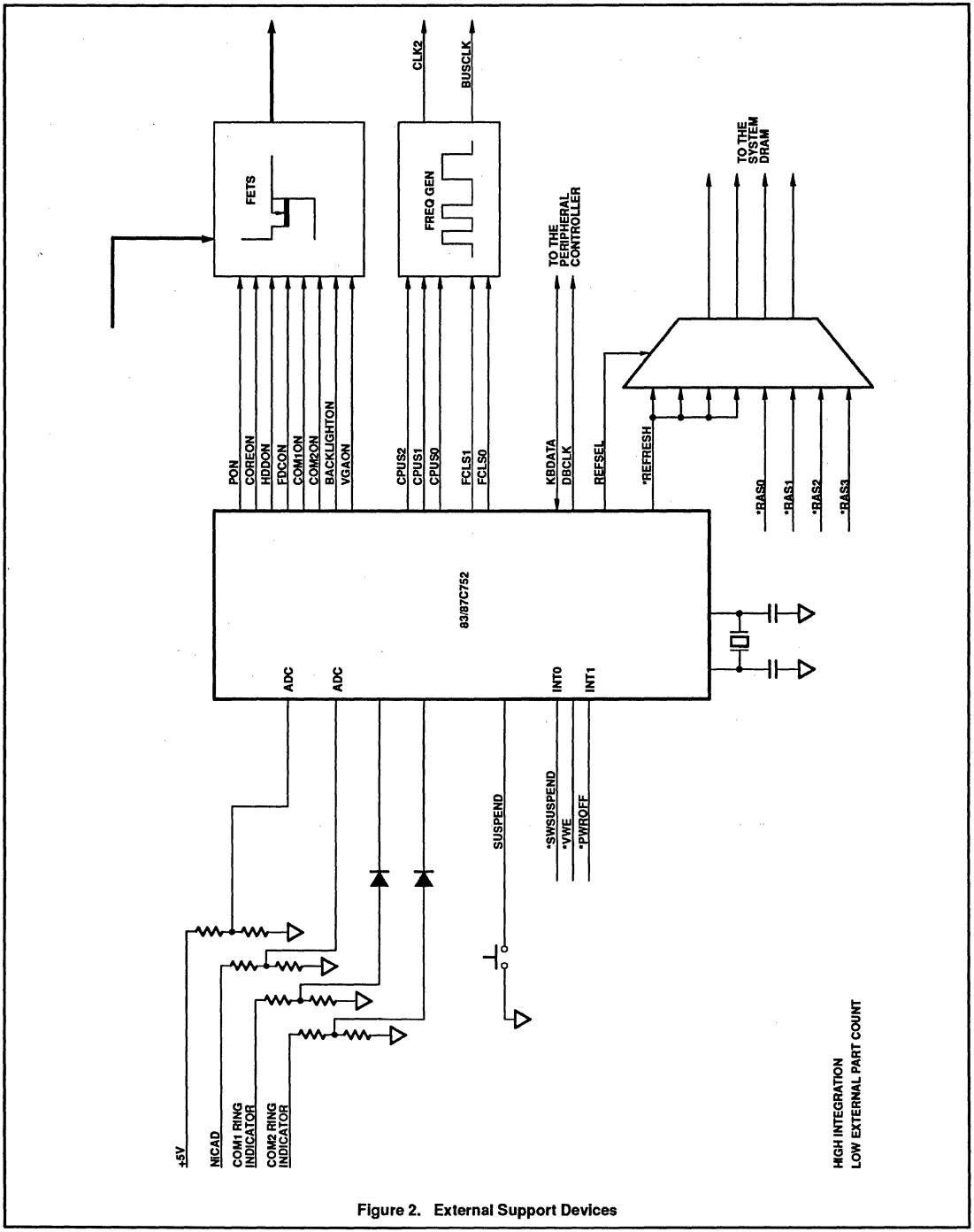


Figure 2. External Support Devices

"Opti-Mizer" power management for notebook computers using the 8XC752 microcontroller

AN436

OPERATION

The power management system operates like a state machine. Transitions from state to state are controlled by expiring timers which are retriggered by external events. On entering a state, external power is switched or clocks are modified. A typical power management system would employ six states: Full Power, Doze, Shutdown, Sleep, Suspend, and Off.

The state diagram of Figure 3 shows the power management states and their interrelationships.

Full Power

Entry to this state is controlled by a transition of the ON/OFF switch. In this state all the power control outputs are asserted, and the clock generator is selected for the highest speed. The system runs at full speed and power.

Doze

This state is entered from the FULL POWER state. Entry into this state is controlled by an expired timer (typically 30 secs). The timer expired as a result of not being reloaded by a transition on an activity monitor input pin. In this state the frequency generator is instructed to reduce the clock speed to about half that of the previous state.

Shutdown

This state is also entered from the FULL POWER state and operates in parallel with the DOZE state. Entry into this state is controlled by an expired timer (typically 30 secs). The timer expired as a result of not being reloaded by a transition on an activity monitor pin. In this state the power to a particular peripheral or group of peripherals is removed via an external FET.

Shutdown-Doze

This is an intermediate state which implements the features of both the SHUTDOWN and DOZE states. Entry into this state from the DOZE state is controlled by an expired timer (typically 30 secs). The timer expired as a result of not being reloaded by a transition on an activity monitor pin. Entry into this state from the SHUTDOWN state is controlled by an expired timer also. The timer expired as a result of not being reloaded by a transition on an activity monitor input pin. In this state the power to a particular peripheral or group of peripherals is removed via an external FET and the frequency generator is instructed to reduce the clock speed to about half that of the FULL POWER state.

Sleep

This state is entered from either the DOZE state or SHUTDOWN state. Entry into this state is controlled by an expired timer (typically 30 secs). The length of this timer is usually longer than that employed in the DOZE or SHUTDOWN states. The timer expired as a result of not being reloaded by a transition on an activity monitor pin. The activity monitor may look for keystrokes or video activity as described below. In this state power is removed from the backlight and LCD modulation voltage regulator via external FETs.

Suspend

This state is entered from any of the above states. Entry into this state is controlled by a transition on an external suspend switch or a command from the BIOS. During this state, the microcontroller takes over the task of refreshing the system memory and removes the power from the rest of the system via external FETs.

Off

This state is entered from any of the above states. Entry into this state is controlled by a transition on an external switch or a command from the BIOS.

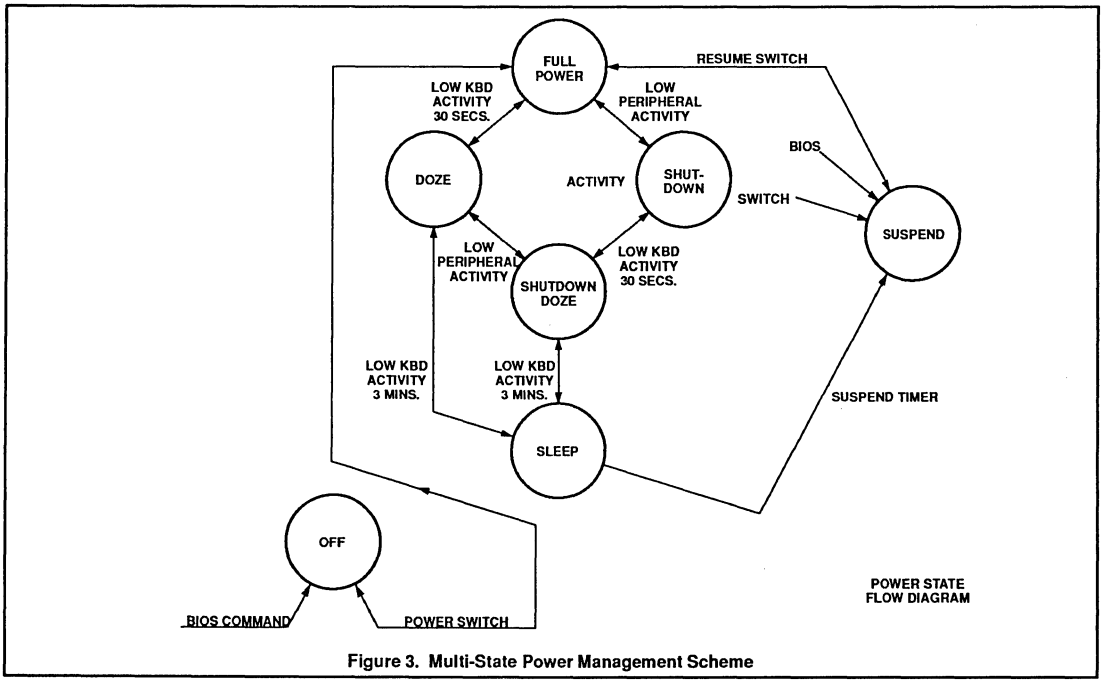


Figure 3. Multi-State Power Management Scheme

"Opti-Mizer" power management for notebook computers using the 8XC752 microcontroller

AN436

POWER MANAGEMENT ELEMENTS

Figure 4 shows the internal power management elements of the Philips/Signetics controller. The Activity monitors contain combinatorial algorithms to monitor or poll an activity or combination of activities. The activity monitors reload programmable timers which toggle clock control and power control output pins.

Each functional block is discussed in more detail below.

Power Outputs

The power outputs are logic level signals that control MOSFETs via level shifting circuitry.

The MOSFETs switch power to the various blocks under power management control and are chosen to have a low R_{ds} on which reduces the voltage drop across the DRAIN-SOURCE channel.

The level shifting circuitry and MOSFET orientation is shown in Figure 5.

C1 and R1 control the switchon edge and should be chosen to make the edge sufficiently slow to minimize the inrush current. This is necessary where the notebook computer employs solid chip tantalum capacitors which fail short-circuit on switchon current transients.

R1 pulls the gate to the NiCAD supply rail. The NiCAD battery voltage is usually greater than +12 Volts. We can exploit this relatively high voltage to turn the MOSFETs hard on and further reduce the R_{ds} on. The NPN transistor is operated as a cascode and gives no net inversion between the logic level and the state of the MOSFET. This is necessary to handle the default powerup mode of the port pins.

For a lower performance and cost reduced system, a logic level FET can be used such as a MTM25N06L and driven directly from the microcontroller port. These logic level FETs usually have R_{ds} on specifications in the region of 100 milliohms. The finite drain-source resistance implies that a small amount of power is wasted in this channel while power is applied to the switched group of devices.

Clock Control

The clock control module controls the speed of the system clocks. Where the notebook system has a synchronous ISA clock, it is derived directly from CLK2, the processor clock. The clock control outputs can be fed directly to a frequency generator such as an AVASEM AV9127 where pins are committed to encode a frequency select scheme. The scheme employs two programmable clock generators; one with eight preset frequencies used for the system clock; and the other with four preset frequencies used for the mass storage subsystem. Figure 6 shows the interconnection between the clock control port and frequency generator.

By changing the assignments of the encoded select lines, the system clock frequency can be reduced. Frequency generators employ analog voltage controlled oscillators which, when instructed to change frequency, will steadily and gradually change frequency in a smooth transition. This scheme does not violate the minimum high and low times for the core logic devices.

Timers

The timers should run independently of the keyboard scanning function. The timers are used as timeouts for a combination of external events or activities. The timers are constructed of reloadable timers and reloaded by transitions or conditions on external events. A typical timeout period is between 1 and 4 minutes, therefore the timeout must be constructed from both timer hardware and support software. Figure 7 shows the interrelationships between hardware and software. The software must record the instances of timeout cycles. If the number of cycles is allowed to reach a predetermined number, the timeout elapses and the assigned power control output is negated, or the clock control outputs proceed to the next state. The count of the number of cycles is reset by a command for the activity monitor.

The activity monitor asserts flags during the background and interrupt tasks. The timer software processes these flags to determine the state of the timeout. The software uses a count variable to measure the instances of the timer elapsing and a flag to determine whether activity has occurred.

Activity Monitor

The activity monitor sets the activity flags for the timers. The monitors contain combinatorial elements which poll an external activity or a number of external activities. External activity can be detected by transitions or levels on input port pins. The interrupt pins would be better suited for transition detect while the general input ports could be used to poll for external conditions.

There are several key activity indicators on the PC. See Table 1 below.

Table 1.

SIGNAL	TYPE	DEVICE/PERIPHERAL
*IDECS1	LEVEL	Hard Disk IDE interface chip select
*IDESC0	LEVEL	Hard Disk IDE interface chip select
*VWE	EDGE	VGA Memory Write enable signal
*FDCS	LEVEL	Floppy disk digital control register
*LPTRDY	LEVEL	Parallel printer interface flag
*GPRD	EDGE	Accessory interface select
*53C90SEL	EDGE	SCSI interface chip select
*LIDSW	LEVEL	Notebook lid switch

"Opti-Mizer" power management for notebook computers using the 8XC752 microcontroller

AN436

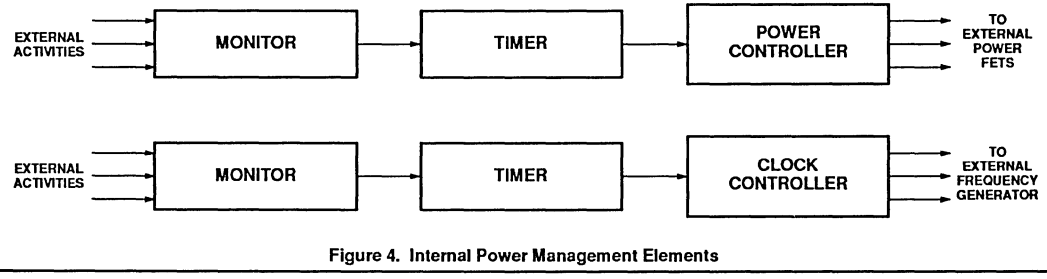


Figure 4. Internal Power Management Elements

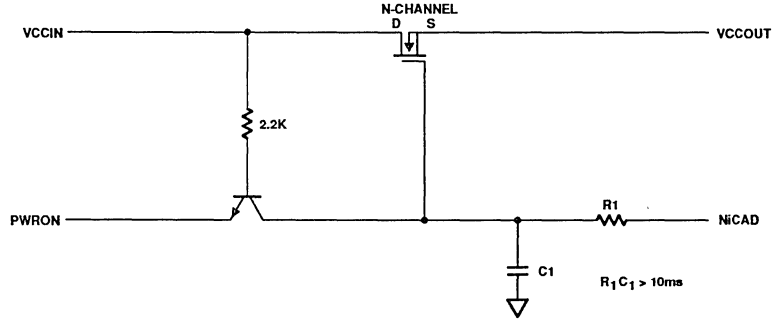


Figure 5. Power Control Circuitry

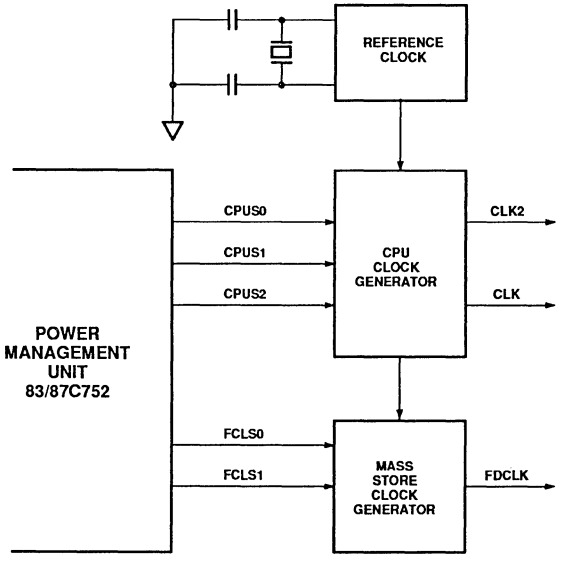


Figure 6. Clock Control Circuitry

"Opti-Mizer" power management for notebook computers using the 8XC752 microcontroller

AN436

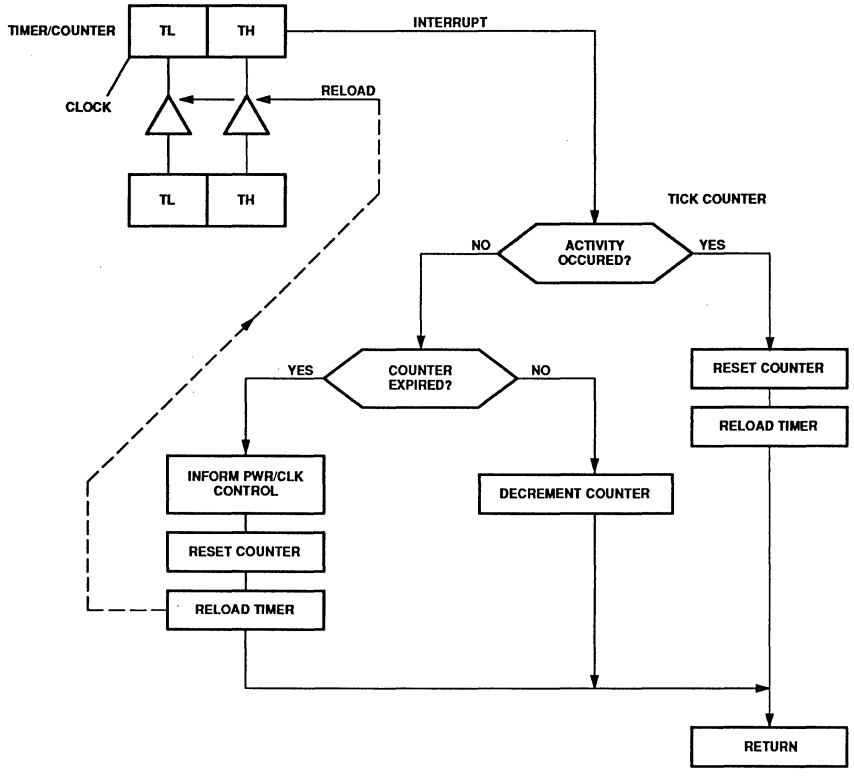


Figure 7. Hardware–Software Interrelationships

Suspend Control

During the suspend state, power is removed from all the system devices except the power management controller and system memory. Before the system is allowed to enter the suspend state, the BIOS empties all the DMA holding registers, makes a copy of the CPU registers and stack pointer into reserved main memory. The VGA memory is also copied into main memory. Once this transaction has been completed, the system BIOS instructs the power management controller to enter the suspend state.

At the start of the suspend state, the microcontroller takes over the memory refresh. This is achieved with the use of an external refresh mux. The mux is switched over to the microcontroller refresh lines when REFSEL is asserted. Switchover must only be executed after the system memory controller has performed a complete refresh cycle (512 / 1024 rows). The external multiplexer asserts all the individual *CAS lines while switched over to the controller. The *RAS lines are all driven by a single *REFRESH command from the controller.

Battery Monitor

The ADC can be used to monitor the battery condition via an external resistor divider. This monitor can be used to assess the condition of the system batteries during system use and while being charged.

A simple "minus delta V" algorithm may be implemented to measure the slope of the battery voltage over time. The charging curve of Figure 8 shows the characteristic of the voltage across the NiCAD cells with a constant current applied. As the NiCADs become charged, the internal temperature of the cells rises and thus their internal resistance increases. This results in a droop of the voltage across the cells. The charging current must be terminated when a droop of greater than 50mV per cell over 1 second is detected.

This minus delta V algorithm can be implemented in the microcontroller software. The analog to digital controller can be used to sample the NiCAD battery voltage level at regular intervals. The sample is compared against the last sample for a negative result. If the magnitude of the result is greater than 50mV per cell, then the charging circuit must be given a command to switch off.

A battery condition detector can also be implemented in the microcontroller. This detector can be used to give an early indication of an exhausted battery. When a NiCAD cell reaches its supply capacity, the voltage across the cell drops rapidly, thus the system supply can only sustain the core logic for a matter of seconds. The battery condition monitor should give the system and user an early indication of this condition. This can be implemented by monitoring the battery voltage. When a preset slope or level is reached or exceeded, the controller can issue an NMI to the core logic. The core logic can

"Opti-Mizer" power management for notebook computers using the 8XC752 microcontroller

execute a shutdown routine which saves the state of the machine on the hard disk and preserves the integrity of the user's data.

PERFORMANCE

Table 2 shows the power performance of the 83/87C752 in a typical notebook system. During each power management state, the core logic system and peripherals demand lower and lower power as the clock speeds are reduced, resets are asserted and power is removed from the device or peripheral.

During the suspend mode, power is completely removed from the core logic devices and peripherals, only the DRAMs remain energized so that the system state can be restored to the next post suspend cycle on the detection of an activity.

The FULL-ON figures are maximums, In reality, the processor executes sporadic scripts and then idles in tight loops awaiting IO. This means that the actual power required by the system under normal conditions will be lower than that estimated in that column of the table.

Table 2.

STATE / DEVICE	FULL-ON POWER (W) (MAX)	SHUTDOWN-DOZE (W)	SLEEP POWER (W)	SUSPEND POWER (W)
Am386SX	2.14	0.7	0.5	0
DRAM + Controller	1.7	0.6	0.4	0.025
Local bus controller	0.3	0.3	0.2	0
ISA bus controller	0.2	0.18	0.07	0
87C752	0.08	0.08	0.08	0.015
BIOS ROM	0.11	0.002	0.002	0
Floppy drive	3.1	0.035	0.035	0
IDE drive	3.3	0.68	0.68	0
VGA controller	2.1	1.6	1.2	0.015
Keyboard controller	0.7	0.6	0.4	0
Keyboard	0.15	0.15	0.15	0
Oscillators	0.1	0.1	0.1	0
RS232 buffers	0.11	0.002	0.002	0
LCD panel	0.7	0.62	0.2	0
Backlight	1.5	1.5	0.1	0
TOTALS	15.69	6.709	4.119	0.055

The gradient of power performance across the states is quite steep, especially when transitioning from the sleep to the suspend states.

Table 3.

	FULL-ON	$^1/_2$ CLK2 SHUTDOWN-DOZE	$^1/_4$ CLK2 SLEEP	SUSPEND
CPU	ON	ON	ON	OFF
DRAM	ON	ON	ON	ON
Local bus	ON	ON	ON	OFF
ISA bus	ON	ON	OFF	OFF
87C752	ON	ON	ON	ON
BIOS ROM	ON	OFF	OFF	OFF
Floppy	ON	ON	OFF	OFF
IDE drive	ON	IDLE	OFF	OFF
VGA	ON	ON	ON	OFF
Keyboard controller	ON	ON	ON	ON
Keyboard	ON	ON	ON	OFF
Oscillators	ON	ON	ON	OFF
RS232 buffers	ON	OFF	OFF	OFF
LCD panel	ON	ON	OFF	OFF
Backlight	ON	ON	OFF	OFF

Note that the panel and backlight are off during the sleep state.

"Opti-Mizer" power management for notebook computers using the 8XC752 microcontroller

AN436

OTHER INTEGRATION OPPORTUNITIES

The 83/87C752 offers a complete power management solution as described above. The functionality and IO capability of this device can be used as a common denominator for other, larger Philips/Signetics microcontrollers. The 83/87C552 offers the same internal functionality while providing more integration capabilities with its increased IO, memory and timer functions.

An example of an integration opportunity would be to combine the keyboard scanner function with the power management function. The Philips/Signetics 83/87C552 microcontroller is ideal for this task. The microcontroller can implement the scanning and code generation schemes associated with the keyboard function, implement the activity monitors, timers and power control scheme required for power management as well as provide an integrated solution to battery condition detection by exploiting the onboard A/D converters.

The PC keyboard scanner is traditionally an 8051 microcontroller. The keyboard scanning and code generation can be performed by an 8051 running at 6MHz. A 12 or 16MHz 83/87C552 microcontroller would have the bandwidth to take on other tasks.

Figure 9 shows the 83/87C552 integrated as a power management unit and keyboard scanner.

Other members of the Philips/Signetics family of 8051 derivative microcontrollers can also provide an integrated solution to power management (see Table 4).

As can be seen, the 83/87C550 provides an intermediate solution to the 83/87C552 and 83/87C752. It has more memory and IO than the 83/87C752 and has three more ADC channels.

Table 4. Microcontrollers with A/D

DEVICE	ROM	RAM	A/D	I/O	PWM	TIMERS
83/87C550	4K	128	8 8-bit	24	0	2
83/87C552	8K	256	8 10-bit	48	2	3
83/87C752	2K	64	5 8-bit	21	1	1

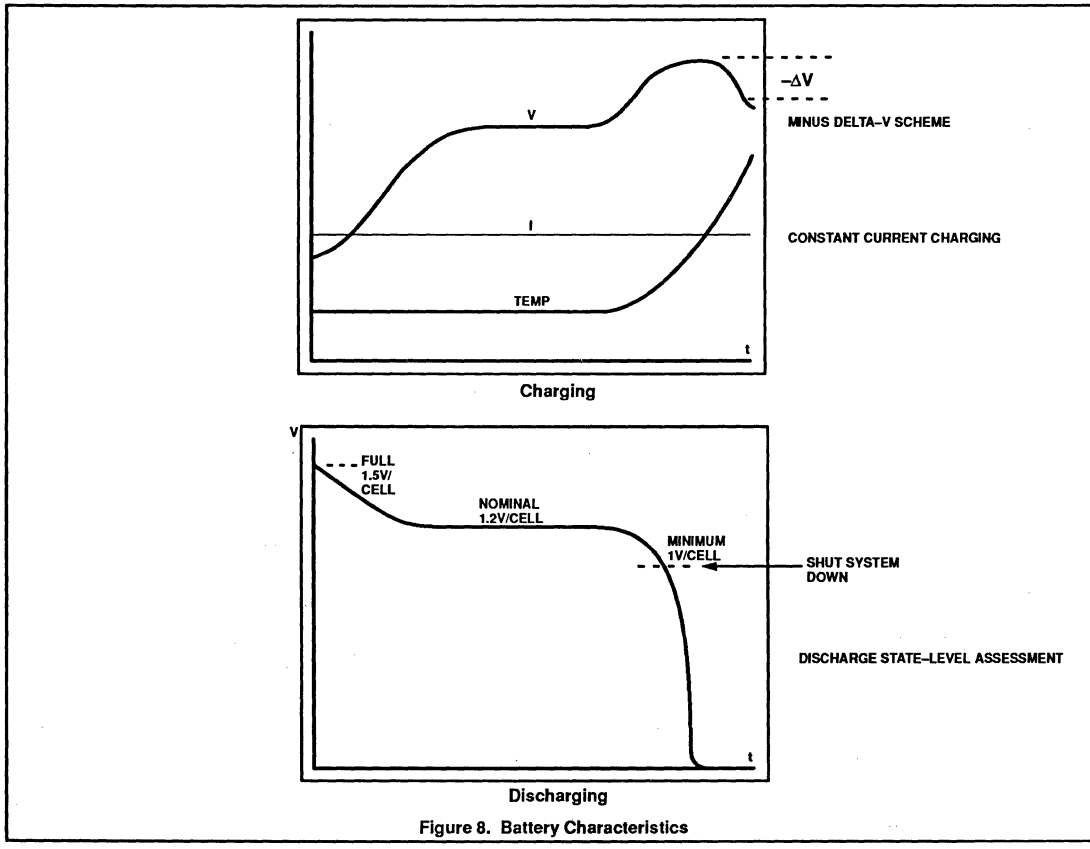


Figure 8. Battery Characteristics

"Opti-Mizer" power management for notebook computers using the 8XC752 microcontroller

AN436

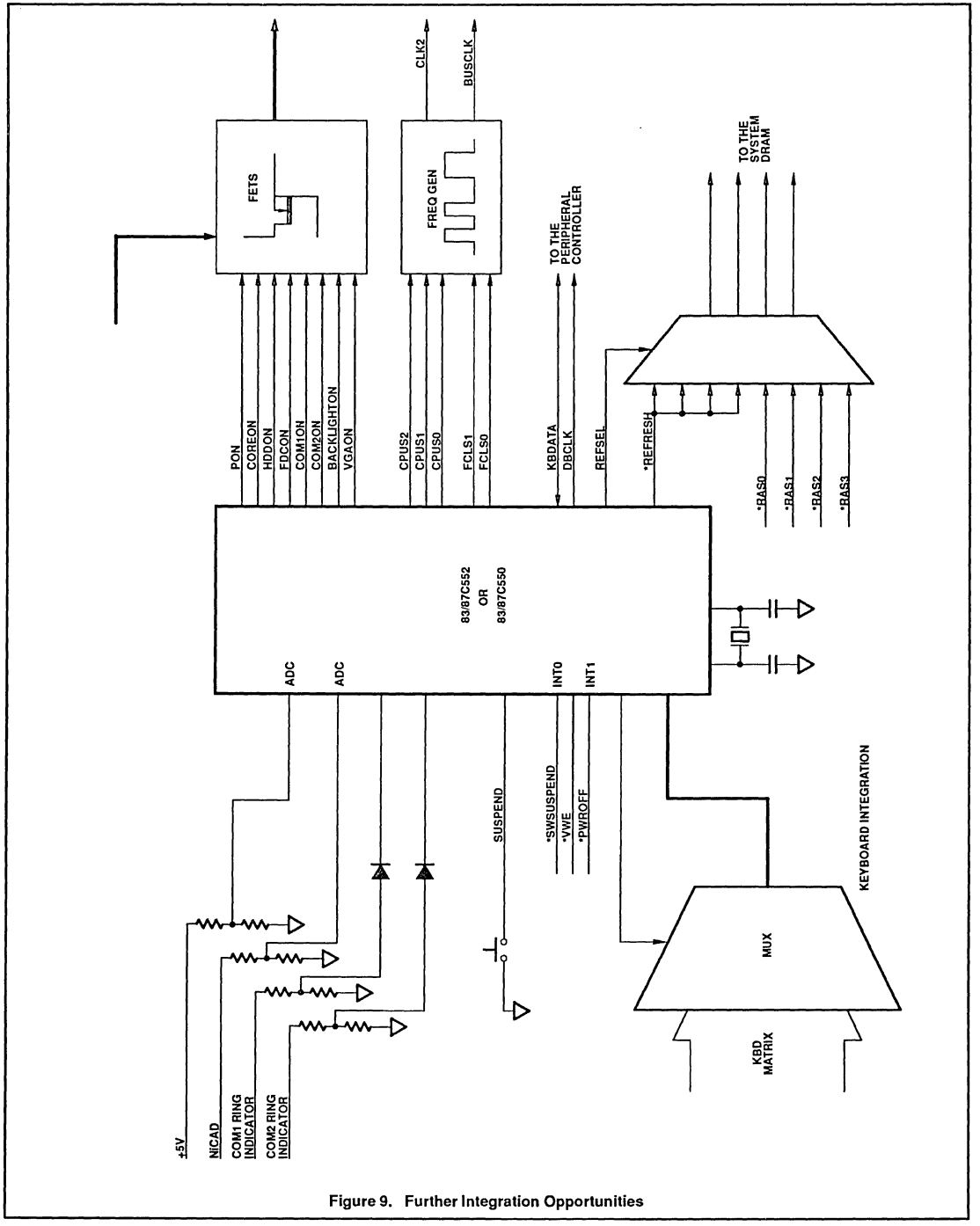
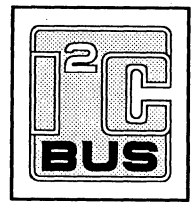


Figure 9. Further Integration Opportunities

I²C routines for 8XC528 AN438

Philips Semiconductors Application Note EIE/AN90015



Summary:

This application note presents a set of software routines, to drive the I²C interface in 8xC528 type of micro controllers. A description of the I²C interface is given. Examples show how to use these routines in PL/M-51, C and assembly source code.

1. INTRODUCTION

This application note describes the I²C interface of the 8xC528 µC and gives a set of routines in application programs to drive this interface.

Chapter 2 gives a hardware description of the bit level I²C. It gives an overview what functions are done in hardware by the interface and the functions that should be implemented by software. The registers described are accessible with software and control the I²C interface.

Chapter 3 gives a description of the routines that may be used by the application program. The routines are written in such a way that the I²C interface becomes transparent to the user. The slave program is descibed in more detail, because this routine may be adapted by the user for his specific application.

Chapter 4 gives simple example programs that show how to use the routines in assembly, PL/M and C application programs.

References:

- The I²C-bus specification; 9398 358 10011
- 8051-based 8-bit Microcontrollers; Data Handbook IC20
- PLM51 I²C Software interface IIC51; ETV/AN89004

2. THE I²C INTERFACE

2.1 CHARACTERISTICS OF I²C INTERFACE

On page 4 the block diagram of the bit-level I²C interface is shown. P1.6/SCL and P1.7/SDA are the serial I/O pins. These two pins meet the I²C specification concerning the input levels and output drive capability. Consequently, these pins have an open drain configuration.
All four modes of the I²C bus can be used:

- Master transmitter
- Master receiver
- Slave transmitter
- Slave receiver

The advantages of using the bit-level I²C hardware compared with a full software implementation are:

- Higher bit rate
- No critical software timing requirements
- Less software overhead
- More reliable data transfer

The bit-level I²C hardware can perform the following functions:

- Filtering the incoming serial data and clock signals. Glitches shorter than 4 Xtal periods are rejected.
- Recognition of a START or STOP condition.
- Generating an interrupt request after reception of a START condition.
- Setting the Bus Busy flag when a START condition is detected.
- Clearing the Bus Busy flag when a STOP condition is detected.
- Recognition of a serial clock pulse on the SCL line.
- Latching the serial data bit on the SDA line at every rising edge on the SCL line.
- Stretching the LOW period of the serial clock SCL to synchronize with external master devices.
- Setting the Read Bit Finished (RBF) or Write Bit Finished (WBF) flag if an error free bit transfer has occurred.
- Setting a Clock LOW-to-HIGH (CLH) flag when a leading edge is detected on the SCL line.
- Generation of serial clock pulse on SCL in master mode.

The following functions must be done with software:

- Handling the I²C interrupt caused by a detected START condition.
- Conversion of serial to parallel data when receiving.
- Conversion of parallel to serial data when transmitting.
- Comparing received slave address with own slave address.
- Interpretation of acknowledge information.
- Guarding the I²C status if the RBF and WBF flags indicate a not regular bit transfer.
- Generating START/STOP conditions when in master mode.
- Handling bus arbitration when in master mode.

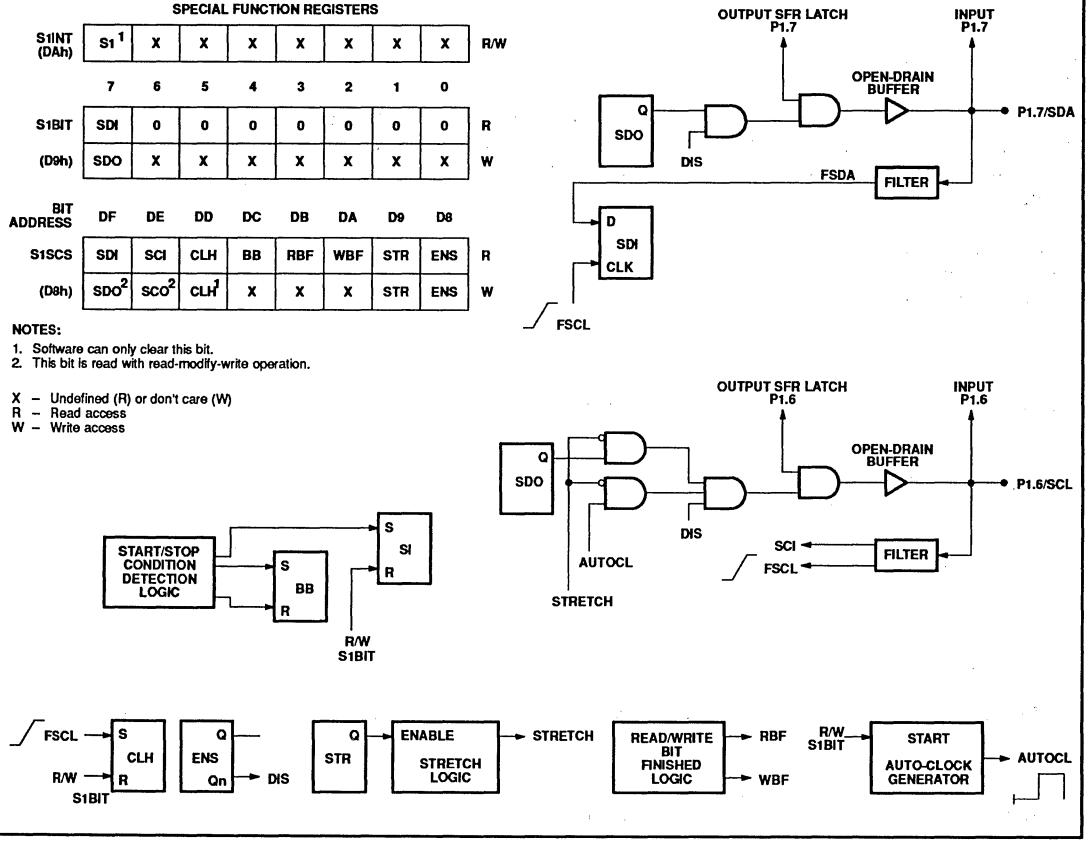
I²C routines for 8XC528 AN438

SPECIAL FUNCTION REGISTERS

S1INT (DAh)	S1[1]	X	X	X	X	X	X	X	R/W
	7	6	5	4	3	2	1	0	
S1BIT	SDI	0	0	0	0	0	0	0	R
(D9h)	SDO	X	X	X	X	X	X	X	W
BIT ADDRESS	DF	DE	DD	DC	DB	DA	D9	D8	
S1SCS	SDI	SCI	CLH	BB	RBF	WBF	STR	ENS	R
(D8h)	SDO[2]	SCO[2]	CLH[1]	X	X	X	STR	ENS	W

NOTES:
1. Software can only clear this bit.
2. This bit is read with read-modify-write operation.

X – Undefined (R) or don't care (W)
R – Read access
W – Write access



2.2 CONTROL AND STATUS REGISTERS

Control of the I²C bus hardware is done via 3 Special Function Registers:
- S1INT: This register contains the serial interrupt flag SI.
- S1BIT: For read this register contains the received bit SDI.
  For write this register contains bit SDO to be transmitted.
- S1SCS: For read this register contains status information.
  For write this register is used as control register.

2.2.1 S1INT: I²C interrupt register

- **S1INT.7** is the Serial Interrupt request flag (SI).
  If the serial I/O is enabled (ENS = 1), then a START condition will be detected and the SI flag is set on the falling edge of the filtered SCL signal.
  Provided that EA (global enable) and ES1 (enable I²C interrupt) are set (in the interrupt enable IE register), SI generates an interrupt that will start the slave address receive routine.
  SI is cleared by accessing the S1BIT register or by writing '00H' to S1INT. SI cannot be set by software

  After reception of a START condition, the LOW period of the SCL pulse is stretched, suspending serial transfer to allow the software to take appropriate action. This clock stretching is ended by accessing the S1BIT register.

2.2.2 S1BIT: Single bit data register

- **S1BIT.7** contains two physical latches; the Serial Data Output (SDO) latch for a write operation and the filtered Serial Data Input (SDI) latch for a read operation. SDI data is latched on the rising edge of the filtered SCL pulse. S1BIT.7 accesses the same physical latches as S1SCS.7, but S1BIT.7 is not bit-addressable.

  Reading or writing S1BIT register starts the next additional actions:
  - SI, CLH, RBF and WBF flags are cleared
  - Stretching the LOW period of the SCL clock is finished.
  - Auto-clock pulse is started if enabled

  The auto-clock is an active HIGH SCL pulse that starts 28 Xtal periods after an access to S1BIT. SCL remains high for 100 Xtal periods. If the SCL line is kept LOW by any device that wants to hold up the bus transfer, the auto-clock counter still runs for 20 Xtal periods to try to make SCL high and then go in a wait-state. This will result in a minimum SCL HIGH time of 80 Xtal periods (5µs at f_{Xtal} = 16MHz).
  The auto-clock signal will be inhibited if the SCO flag in the S1SCS register is set to '1'. SCL pulses must then be generated by software. In this situation access to S1BIT may be used to clear the SI, CLH, RBF and WBF flags.

  A quick check on a successful bit transfer from/to SDO/SDI is carried out by testing only the RBF or WBF flag (see 2.2.3).

2.2.3 S1SCS: Control and status register

- **S1SCS.7** represents two physical latches, the Serial Data Output (SDO) latch for write operations and the Serial Data Input (SDI) latch for read operations. S1SCS.7 accesses the same physical latches as S1BIT.7, but S1SCS.7 is bit addressable. However a read or write operation of S1SCS.7 does not start an auto-clock pulse, will not finish clock stretching and will not clear flags!

- **S1SCS.6** represents two physical latches, the Serial Clock Output (SCO) latch for write operations and the Serial Clock Input (SCI) latch for read operations. The output of SCO is "OR-ed" with the auto-clock pulse. If SCO = '1' the auto-clock generation is disabled and its output is LOW. Internal clock stretching logic and external devices then can pull the SCL line LOW.
  If the auto-clock is not used the SCL line has to be controlled by setting SCO = '1', waiting for CLH to become '1' and setting SCO = '0' after the specified SCL HIGH time. Data access should be done via S1SCS.7.

- **S1SCS.5** is the serial Clock LOW-to-HIGH transition flag (CLH). This flag is set by a rising edge of the filtered serial clock. CLH = '1' indicates that no devices are stretching SCL LOW, and since the last CLH reset, a new valid data bit has been latched in SDI.
  CLH can be cleared by writing '0' to S1SCS.5 or by a read or write operation to the S1BIT register. Clearing CLH also clears RBF and WBF. Writing a '1' to S1SCS.5 will not affect CLH.

- **S1SCS.4** is the Bus Busy flag (BB). BB is set or cleared by hardware only. If set it indicates that a START condition has been detected on the I²C bus. A STOP condition clears the BB-flag.

- **S1SCS.3** is the Read Bit Finished flag (RBF). If RBF = 1 it indicates that a serial bit has been received and latched into SDI successfully. If during a bit transfer RBF is '0', the cause is indicated as follows:
  - SCI = '1' and CLH = '1': The SCL pulse is not finished and still HIGH.
  - CLH = '0': A bus device is delaying the transfer by stretching the LOW level on the SCL line.
  - BB = '0': A STOP-condition has been detected during the bit transfer. This should be considered as a bus-error.
  - SI = '1': A START-condition has been detected during the bit transfer. This should be considered as a bus-error.

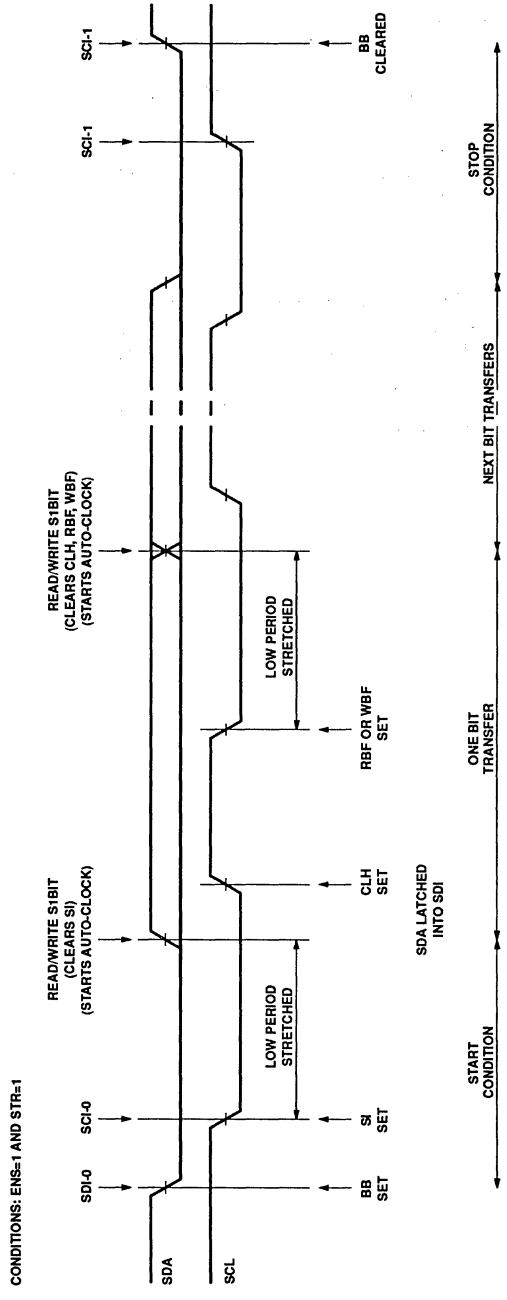
  RBF can be cleared by clearing CLH or by a read or write operation to the S1BIT register.

- **S1SCS.2** is the Write Bit Finished flag (WBF). If set it indicates that a serial bit in SDO has been transmitted successfully. If during bit transfer WBF is '0', the following conditions may be the cause:
  - SCI = '1' and CLH = '1': The SCL pulse is not finished and still HIGH.
  - CLH = '0': A bus device is delaying the transfer by stretching the LOW level on the SCL line.
  - BB = '0': A STOP-condition has been detected during the bit transfer. This should be considered as a bus-error.
  - SI = '1': A START-condition has been detected during the bit transfer. This should be considered as a bus-error.

  WBF can be cleared by clearing CLH or access to the S1BIT register.

S1SCS.1 is the STRetch control flag (STR). STR can be set or cleared by software only. Setting STR enables the stretching of SCL LOW periods. Stretching will occur after a falling edge on the filtered serial clock. This allows synchronization with the SCL clock signal of an external master device.
If STR is cleared, no stretching of the SCL LOW period will occur after the transfer of a serial bit.
The LOW level on the SCL line is also stretched after a START condition is received, regardless of the STR contents. The stretching of the SCL LOW period is finished by a read or write operation of the S1BIT register.

S1SCS.0 is the ENable Serial I/O flag (ENS).
ENS can be set or cleared by software only.
ENS = '0' disables the serial I/O. The I/O signals P1.6/SCL and P1.7/SDA are determined by the port latches of P1.6 and P1.7 (open drain). If P1.6 and P1.7 are connected to an I²C bus, then the flags SDI, SCI, CLH and BB still monitor the I²C bus status, but will not influence the I/O lines, nor will they request an interrupt.
ENS='1' enables the START detection and clock stretching logic. Note that the P1.6 and P1.7 latches and the SDO and SCO control flags must be set to '1' before ENS is set to avoid SCL and/or SDA to pull the lines LOW.



Example of a Serial Transfer

3: I²C ROUTINES

3.1 INTRODUCTION

A set of routines is written for the I²C interface that supports multi-master and slave operation. The routines are placed in a library I2C_DR.LIB. If I2C_DR.LIB is linked to an application program, only the needed object modules are linked in the output file.
The routines can be used as device driver for PL/M-51, C and 8051-assembly code. By using these routines the bit-level I²C interface is fully transparent for the user.

The routines use the following 8xC528 resources:
- Exclusive use of Register_Bank_1. Only R7 of this register bank contains static data (Own Slave Address). R0..R6 may be used by the application program when the I²C routine is finished.
- 7 bytes DATA used for parameter passing.
- 1 byte Bit-Addressable DATA for status flags.

When using routines from this library DPH, DPL, PSW (except CY) and B are not altered.

An n-bytes data buffer is used as destination or source buffer for the bytes to be received/transmitted and reside in DATA or IDATA memory space.

The code is written to generate the highest transfer rate on the I²C bus. At f_{Xtal} = 16Mhz this will result in a bit rate of 87.5kbit/sec.

The following software tools from Tasking/BSO are used for program development:
- OM4142 Cross Assembler 8051 for DOS: V3.0b
- OM4144 PL/M 8051 Compiler for DOS: V3.0a
- OM4136 C8051 Compiler for DOS: V1.1a
- OM4129 XRAY51 debugger: V1.4c

3.2 FUNCTIONAL DESCRIPTION

When using these routines in a PL/M application program they must be declared EXTERNAL. In this declaration the user can specify the type returned by each procedure. All procedures (except Init_IIC and Dis_IIC) can return a BIT or BYTE, depending on the chosen EXTERNAL declaration. The BIT or BYTE returned is '0' if the I²C was successful. If a BYTE is returned the following check bits are available for the user:

BYTE.0: An I²C error has been detected.
BYTE.1: No ACK received.
BYTE.2: Arbitration lost.
BYTE.3: Time out error. This may be caused by an external device pulling SCL LOW.
BYTE.4: A bus error has occured. This may be a spurious START/STOP during a bit transfer.
BYTE.5: No access to I²C bus.
BYTE.6: 0
BYTE.7: 0

Note that typed procedures must be called using an expression. If the result of an I²C procedure is to be ignored, a dummy assignment must be done for a typed procedure. The examples in the following section assume that the procedures are called from a PL/M program. Examples will be given later how to use these routines with C and assembly application programs.

3.2.1 Init_IIC

Declaration:

```
Init_IIC:
   PROCEDURE (Own_Slave_Address, Slave_Sub_Address) EXTERNAL;
   DECLARE   (Own_Slave_Address, Slave_Sub_Address) BYTE;
   END;
```

Description:

Init_IIC must be called after RESET, before any procedure is called. The I²C interface and I²C interrupt will be enabled. The global enable interrupt flag however will not be effected. This should be done afterwards. Own_Slave_Address is passed to Init_IIC for use as slave. Slave_Sub_Address is the pointer to a DATA buffer that is used for data transfer in slave mode. When used as master in a single master system, these parameters are not used.

Example:

```
CALL Init_IIC (54h,.Slave_Data_Buffer);
ENABLE; /* Enable Interrupts; EA=1 */
```

3.2.2 Dis_IIC

Declaration:

```
Dis_IIC:
   PROCEDURE EXTERNAL;
```

Description:

Dis_IIC will disable the I²C-interface and the I²C-interrupt.
The I²C interface will still monitor the bus, but will not influence the SDA and SCL lines.

Example:

```
CALL Dis_IIC;
```

3.2.3 IIC_Test_Device

Declaration:

```
IIC_Test_Device:
   PROCEDURE (Slave_Address) [BIT|BYTE] EXTERNAL;
   DECLARE   (Slave_Address) BYTE;
   END;
```

Description:

IIC_Test_Device just sends the slave address to the I²C bus. It can be used to check the presence of a device on the I²C bus

I²C Protocol:

S-SlvW-A-P : Device is present, IIC_Error=0
S-SlvW-N-P : Device is not present, IIC_Error=1

Example:

```
DECLARE IIC_Error BIT;
.....
IIC_Error=IIC_Test_Device(8Ch);
IF (IIC_Error) THEN
    "Device not acknowledging on slave address"
ELSE
    "Device acknowledges on slave address"
```

3.2.4 IIC_Write

Declaration:

```
IIC_Write:
   PROCEDURE (Slave_Address, Count, Source_Ptr) [BIT|BYTE] EXTERNAL;
   DECLARE   (Slave_Address, Count, Source_Ptr) BYTE;
   END;
```

Description:

IIC_Write is the most basic procedure to write a message to a slave device.

I²C Protocol:

```
L           =Count
D1[0..L-1]  BASED by Source_Ptr

S-SlvW-A-D1[0]-A....A-D1[L-1]-A-P
```

Example:

```
DECLARE Data_Buffer(4) BYTE;
.....
CALL IIC_Write(02Ch, LENGTH(Data_Buffer),.Data_Buffer);
```

3.2.5 IIC_Write_Sub

Declaration:

```
IIC_Write_Sub:
   PROCEDURE (Slave_Address, Count, Source_Ptr, Sub_Address) [BIT|BYTE] EXTERNAL;
   DECLARE   (Slave_Address, Count, Source_Ptr, Sub_Address) BYTE;
   END;
```

Description:

IIC_Write_Sub writes a message preceded by a sub-address to a slave device

I²C Protocol:

```
L          =Count
Sub        =Sub_Address
D1[0..L-1] BASED by Source_Ptr

S-SlvW-A-Sub-A-D1[0]-A-D1[1]-A....A-D1[L-1]-A-P
```

Example:

```
DECLARE Data_Buffer(8) BYTE;
.....
CALL IIC_Write_Sub (48h,LENGTH(Data_Buffer),.Data_Buffer,2);
```

3.2.6 IIC_Write_Sub_SWInc

Declaration:

```
IIC_Write_Sub_SWInc:
   PROCEDURE (Slave_Address, Count, Source_Ptr, Sub_Address) [BIT|BYTE] EXTERNAL;
   DECLARE   (Slave_Address, Count, Source_Ptr, Sub_Address) BYTE;
   END;
```

Description:

Some I²C devices addressed with a sub-address do not automatically increment the sub-address after reception of each byte. IIC_Write_Sub_SWInc can be used for such devices the same way as IIC_Write_Sub is used. IIC_Write_Sub_SWInc splits up the message in smaller messages and increments the sub-address itself.

I²C Protocol:

```
L           =Count
Sub         =Sub_Address
D1[0..L-1]  BASED by Source_Ptr

S-SlvW-A- (Sub+0) - A-D1[0] - A-P
S-SlvW-A- (Sub+1) - A-D1[1] - A-P
.....
S-SlvW-A- (Sub+L-1)-A-D1[L-1]-A-P
```

Example:

```
DECLARE Data_Buffer(6) BYTE;
.....
CALL IIC_Write_Sub_SWInc(80h,LENGTH(Data_Buffer),.Data_Buffer,2);
```

3.2.7 IIC_Write_Memory

Declaration:

```
IIC_Write_Memory:
   PROCEDURE (Slave_Address, Count, Source_Ptr, Sub_Address) [BIT|BYTE] EXTERNAL;
   DECLARE   (Slave_Address, Count, Source_Ptr, Sub_Address) BYTE;
   END;
```

Description:

I²C Non-Volatile Memory devices (such as PCF8582) need an additional delay after writing a byte to it. IIC_Write_Memory can be used to write to such devices the same way IIC_Write_Sub is used. IIC_Write_Memory splits up the message in smaller messages and increments the sub-address itself. After transmission of each message a delay of 40 milliseconds (f_{Xtal} = 16MHz) is inserted.

I²C Protocol:

```
L          =Count
Sub        =Sub_Address
D1[0..L-1] BASED by Source_Ptr

S-SlvW-A- (Sub+0)  -  A- D1[0] - A-P
       Delay 40ms
S-SlvW-A- (Sub+1)  -  A- D1[1] - A-P
       Delay 40ms
.....
S-SlvW-A- (Sub+L-1)-  A- D1[L-1]-A-P
       Delay 40ms
```

Example:

```
DECLARE Data_Buffer(10) BYTE;
.....
CALL IIC_Write_Memory(0A0h,LENGTH(Data_Buffer),.Data_Buffer,0F0h);
```

3.2.8 IIC_Write_Sub_Write

Declaration:

```
IIC_Write_Sub_Write:
   PROCEDURE (Slave_Address, Count1, Source_Ptr1, Sub_Address, Count2, Source_Ptr2)
                                                    [BIT|BYTE] EXTERNAL;
   DECLARE   (Slave_Address, Count1, Source_Ptr1, Sub_Address, Count2, Source_Ptr2)
                                                                BYTE;
   END;
```

Description:

IIC_Write_Sub_Write writes 2 data blocks preceded by a sub-address in one message to a slave device. This procedure can be used for devices that need an extended addressing method, without the need to put all data into one large buffer. Such a device is the ECCT (I²C controlled teletext device; see example).

I²C Protocol:

```
L          =Count1
M          =Count2
Sub        =Sub_Address
D1[0..L-1] BASED by Source_Ptr1
D2[0..M-1] BASED by Source_Ptr2

S-SlvW-A-Sub-A-D1[0]-A-D1[1]-A-....-A-D1[L-1]-A-D2[0]-A-D2[1]-A-....-A-D2[M-1]-A-P
```

Example:

```
PROCEDURE Write_CCT_Memory (Chapter, Row, Column, Data_Buf, Data_Count);
DECLARE (Chapter, Row, Column, Data_Buf, Data_Count) BYTE;

/*
   The extended address (CCT-Cursor) is formed by Chapter, Row and Column. These three
   bytes are written after the sub-address (=8) followed by the actual data that will be stored
   relative to the extended address.
*/

CALL IIC_Write_Sub_Write (22h, 3, .Chapter, 8, Data_Buf, Data_Count);
END Write_CCT_Memory;
```

3.2.9 IIC_Write_Sub_Read

Declaration:

```
IIC_Write_Sub_Read:
   PROCEDURE (Slave_Address, Count1, Source_Ptr1, Sub_Address, Count2, Dest_Ptr2)
                                                          [BIT|BYTE] EXTERNAL;
   DECLARE   (Slave_Address, Count1, Source_Ptr1, Sub_Address, Count2, Dest_Ptr2)
                                                                 BYTE;
   END;
```

Description:

IIC_Write_Sub_Read writes a data block preceeded by a sub-address, generates an I²C restart condition, and reads a data block. This procedure can be used for devices that need an extended addressing method. Such a device is the ECCT.

I²C Protocol:

```
L          =Count1
M          =Count2
Sub        =Sub_Address
D1[0..L-1] BASED by Source_Ptr1
D2[0..M-1] BASED by Source_Ptr2

S-SlvW-A-Sub-A-D1[0]-A-D1[1]-A-....-A-D1[L-1]-A-S-SlvR-A-D2[0]-A-D2[1]-A-....-A-D2[M-1]-N-P
```

Example:

```
PROCEDURE Read_CCT_Memory (Chapter, Row, Column, Data_Buf, Data_Count);
DECLARE (Chapter, Row, Column, Data_Buf, Data_Count) BYTE;

/*
   The extended address (CCT-Cursor) is formed by Chapter, Row and Column. These three
   bytes are written after the sub-address (8). After that the actual data will be read relative to
   the extended address.
*/

CALL IIC_Write_Sub_Write (22h, 3, .Chapter, 8, Data_Buf, Data_Count);
END Read_CCT_Memory;
```

3.2.10 IIC_Write_Com_Write

Declaration:

```
IIC_Write_Com_Write:
   PROCEDURE (Slave_Address, Count1, Source_Ptr1, Count2, Source_Ptr2) [BIT|BYTE]
                                                                  EXTERNAL;
   DECLARE   (Slave_Address, Count1, Source_Ptr1, Count2, Source_ptr2) BYTE;
   END;
```

Description:

IIC_Write_Com_Write writes two data blocks from different data buffers in one message to a slave receiver. This procedure can be used for devices where the message consists of 2 different data blocks. Such devices are for instance LCD-drivers, where the first part of the message consists of addressing and control information, and the second part is the data string to be displayed.

I²C Protocol:

```
L          =Count1
M          =Count2
D1[0..L-1] BASED by Source_Ptr1
D2[0..M-1] BASED by Source_Ptr2

S-SlvW-A-D1[0]-A-D1[1]-A-....-A-D1[L-1]-A-D2[0]-A-D2[1]-A-....-A-D2[M-1]-A-P
```

Example:

```
DECLARE Control_Buffer(2) BYTE;
DECLARE Data_Buffer(20) BYTE;
.....
CALL IIC_Write_Com_Write(74h, LENGTH(Control_Buffer), .Control_Buffer,
                              LENGTH(Data_Buffer), .Data_Buffer);
```

3.2.11 IIC_Write_Rep_Write

Declaration:

```
IIC_Write_Rep_Write:
  PROCEDURE (Slave_Address1, Count1, Source_Ptr1, Slave_Address2, Count2, Source_ptr2)
                                                          [BIT|BYTE] EXTERNAL;
  DECLARE   (Slave_Address1, Count1, Source_ptr1, Slave_Address2, Count2, Source_Ptr2)
                                                                  BYTE;
  END;
```

Description:

Two data strings are sent to separate slave devices, separated with a repeat START condition. This has the advantage that the bus does not have to be released with a STOP condition before the transfer from the second slave.

I²C Protocol:

```
L          =Count1
M          =Slave_Address2
SlvW1      =Slave_Address1
SlvW2      =Slave_Address2
D1[0..L-1] BASED by Source_Ptr1
D2[0..M-1] BASED by Source_Ptr2

S-SlvW-A-D1[0]-A-D1[1]-....-A-D1[L-1]-A-S-SlvW-A-D2[0]-A-D2[1]-....-A-D2[M-1]-A-P
```

Example:

```
DECLARE Data_Buffer_1(10) BYTE;
DECLARE Data_Buffer_2(4) BYTE;
.....
CALL IIC_Write_Rep_Write (48h, LENGTH(Data_Buffer_1), .Data_Buffer_1, 50h,
                          LENGTH(Data_Buffer_2), .Data_Buffer_2);
```

3.2.12 IIC_Write_Rep_Read

Declaration:

```
IIC_Write_Rep_Read:
  PROCEDURE (Slave_Address1, Count1, Source_Ptr1, Slave_Address2, Count2, Dest_ptr2)
                                                        [BIT|BYTE] EXTERNAL;
  DECLARE   (Slave_Address1, Count1, Source_ptr1, Slave_Address2, Count2, Dest_Ptr2) BYTE;
  END;
```

Description:

A data string is sent and received to/from two separate slave devices, separated with a repeat START condition. This has the advantage that the bus does not have to be released with a STOP condition before the transfer from the second slave.

I²C Protocol:

```
L          =Count1
M          =Count2
SlvW1      =Slave_Address1
SlvW2      =Slave_Address2
D1[0..L-1] BASED by Source_Ptr1
D2[0..M-1] BASED by Dest_Ptr2

S-SlvW-A-D1[0]-A-D1[1]-....-A-D1[L-1]-A-S-SlvR-A-D2[0]-A-D2[1]-....-A-D2[M-1]-N-P
```

Example:

```
DECLARE Data_Buffer_1(10) BYTE;
DECLARE Data_Buffer_2(4) BYTE;
.....
CALL IIC_Write_Rep_Read (48h, LENGTH(Data_Buffer_1), .Data_Buffer_1, 57h,
                         LENGTH(Data_Buffer_2), .Data_Buffer_2);
```

3.2.13 IIC_Read

Declaration:

```
IIC_Read:
   PROCEDURE (Slave_Address, Count, Dest_Ptr) [BIT|BYTE] EXTERNAL;
   DECLARE   (Slave_Address, Count, Dest_Ptr) BYTE;
   END;
```

Description:

IIC_Read is the most basic procedure to read a message from a slave device.

I²C Protocol:

```
M           =Count
D2[0..M-1]  BASED by Dest_Ptr

S-SlvR-A-D2[0]-A-D2[1]-A....-A-D2[M-1]-N-P
```

Example:

```
DECLARE Data_Buffer(4) BYTE;
.....
CALL IIC_Read (0B5, LENGTH(Data_Buffer), .Data_Buffer);
```

3.2.14 IIC_Read_Status

Declaration:

```
IIC_Read_Status:
   PROCEDURE (Slave_Address, Dest_Ptr) [BIT|BYTE] EXTERNAL;
   DECLARE   (Slave_Address, Dest_Ptr) BYTE;
   END;
```

Description:

Several I²C devices can send a one byte status-word via the bus. IIC_Read_Status can be used for this purpose. IIC_Read_Status works the same way as IIC_Read but the user does not have to pass a count parameter.

I²C Protocol:

```
Status      BASED by Dest_Ptr

S-SlvR-A-Status-N-P
```

Example:

```
DECLARE Status_Byte BYTE;
.....
CALL IIC_Read_Status (84h, .Status_Byte);
```

3.2.15 IIC_Read_Sub

Declaration:

```
IIC_Read_Sub:
   PROCEDURE (Slave_Address, Count, Dest_Ptr, Sub_Address) [BIT|BYTE] EXTERNAL;
   DECLARE   (Slave_Address, Count, Dest_Ptr, Sub_Address) BYTE;
   END;
```

Description:

IIC_Read_Sub reads a message from a slave device, preceeded by a write of the sub-address. Between writing the sub-address and reading the message an I²C restart condition is generated without releasing the bus. This prevents other masters from accessing the slave device in between and overwriting the sub-address.

I²C Protocol:

```
M          =Count
Sub        =Sub_Address
D2[0..M-1] BASED by Dest_Ptr

S-SlvW-A-Sub-A-S-SlvR-D2[0]-A-D2[1]-A....A-D2[M-1]-N-P
```

Example:

```
DECLARE Data_Buffer(5) BYTE;
.....
CALL IIC_Read_Sub (0A3h, LENGTH(Data_Buffer), .Data_Buffer, 2);
```

3.2.16 IIC_Read_Rep_Read

Declaration:

```
IIC_Read_Rep_Read:
   PROCEDURE (Slave_Address1, Count1, Dest_Ptr1, Slave_Address2, Count2, Dest_ptr2)
                                                          [BIT|BYTE] EXTERNAL;
   DECLARE   (Slave_Address1, Count1, Dest_ptr1, Slave_Address2, Count2, Dest_Ptr2) BYTE;
   END;
```

Description:

Two data strings are read from separate slave devices, separated with a repeat START condition. This has the advantage that the bus does not have to be released with a STOP condition before the transfer from the second slave.

I²C Protocol:

```
L          =Count1
M          =Count2
SlvW1      =Slave_Address1
SlvW2      =Slave_Address2
D1[0..L-1] BASED by Dest_Ptr1
D2[0..M-1] BASED by Dest_Ptr2

S-SlvR-A-D1[0]-A-D1[1]-....-A-D1[L-1]-N-S-SlvR-A-D2[0]-A-D2[1]-....-A-D2[M-1]-N-P
```

Example:

```
DECLARE Data_Buffer_1(10) BYTE;
DECLARE Data_Buffer_2(4) BYTE;
.....
CALL IIC_Read_Rep_Read (49h, LENGTH(Data_Buffer_1), .Data_Buffer_1, 51h,
LENGTH(Data_Buffer_2), .Data_Buffer_2);
```

3.2.17 IIC_Read_Rep_Write

Declaration:

```
IIC_Read_Rep_Write:
   PROCEDURE (Slave_Address1, Count1, Dest_Ptr1, Slave_Address2, Count2, Source_ptr2)
                                                          [BIT|BYTE] EXTERNAL;
   DECLARE   (Slave_Address1, Count1, Dest_ptr1, Slave_Address2, Count2, Source_Ptr2) BYTE;
   END;
```

Description:

A data string is received and send from/to two separate slave devices, separated with a repeat START condition. This has the advantage that the bus does not have to be released with a STOP condition before the transfer from the second slave.

I²C Protocol:

```
L          =Count1
M          =Count2
SlvW1      =Slave_Address1
SlvW2      =Slave_Address2
D1[0..L-1] BASED by Dest_Ptr1
D2[0..M-1] BASED by Source_Ptr2

S-SlvR-A-D1[0]-A-D1[1]-....-A-D1[L-1]-N-S-SlvW-A-D2[0]-A-D2[1]-....-A-D2[M-1]-A-P
```

Example:

```
DECLARE Data_Buffer_1(10) BYTE;
DECLARE Data_Buffer_2(4) BYTE;
.....
CALL IIC_Read_Rep_Write (49h, LENGTH(Data_Buffer_1), .Data_Buffer_1, 58h,
                         LENGTH(Data_Buffer_2), .Data_Buffer_2);
```

3.2.18 Slave mode routines

There are two ways for the I²C interface to enter the slave-mode:
- After an I²C interrupt the software must enter the slave-receiver mode to receive the slave address. This address will then be compared with its own address. If there is a match either slave-transmitter or slave-receiver mode will be entered. If no match occurs, the interrupted program will be continued.
- During transmission of a slave-address in master-mode, arbitration is lost to another master. The interface must then switch to slave-receiver mode to check if this other master wants to address the 8xC528 I²C interface.

The slave-mode protocol is very application dependent. In this note the basic slave-receive and slave-transmit routines are given and should be considered as examples. The user may for instance send NO_ACK after receiving a number of bytes to signal to the master-transmitter that a data buffer is full. A description of the code will be given later.

Slave parameters are given with the Init_IIC procedure. The passed parameters are the own-slave-address and a source/destination-pointer to a data buffer.

The slave-routine will be suspended at the following conditions:
- Interrupts with higher priority. Slave-routine will be resumed again after interrupt is handled.
- If a NO_ACKNOWLEDGE is received from a master-receiver.
- If a STOP condition is detected from a master transmitter.

Constraints for user software.
- The user must control the global enable (EA) bit.
- The user must control the priority level of the I²C interrupt. If the slave routine is interrupted by a higher priority interrupt, the SCL line will be stretched to postpone bus transfer until the higher interrupt is finished.

3.3 THE SLAVE ROUTINE: SLAVE.ASM

On page 30 the listing of the slave routine can be seen.
The routine is written in such a way that stretching is of SCL is minimized. Application code can be inserted in this routine and this will increase stretching time.

The routine has 2 entry points.

Entry via MST_ENTRY happens when an arbitration error has occurred when transmitting a slave address in master mode. Auto-clock generation will be disabled and SCL stretching enabled. The byte will be continued to be received and can later be compared with the own slave address.

The second entry point is via an interrupt when a START condition is detected. At __PIP0A the context of the interrupted program is stored. Next Auto-clock generation is disabled and SCL stretching enabled. Reception of the slave address can now begin by calling RCV_SL_BY. When the received slave-address is compared with the own-slave-address the R/W-bit is ignored. If there is no match between the 2 addresses, a negative ACK bit is sent and the slave routine is left via EXIT. If there was a match the R/W bit is checked to enter the slave-receiver or slave-transmitter mode.

The slave-transmitter mode starts at NXT_TRX. After getting the byte from the data buffer via BUF_POINT and initialising the bit counter BIT_CNT the transmission loop is entered. A bit is written via access to S1BIT because this will automatically reset the CLH and WBF status flags, and also SCL stretching. Now WBF must be tested until the transmission is successful. When WBF becomes true SCL will be stretched again. When 8 bits are sent the SDA line is released and RBF is tested until the ACK bit is received. The ACK bit is read by reading SDI in stead of S1BIT to maintain SCL stretching. If ACK was false no more bytes have to be sent and the routine is left. If another byte has to be transmitted, BUF_POINT is updated and transmission will continue.

The slave-receiver mode starts at RCV_SLAVE. A byte is received by calling RCV_SL_BY. This routine will clear the CY-flag when a STOP condition has been received. This means that the master will send no more bytes to this slave and the slave routine will be left. When no STOP condition was detected the received byte will be stored @BUF_POINT and an ACK bit will be sent. After this a new byte can be received.
When calling RCV_SL_BY the bit counter BIT_CNT will be initialized and the SCL stretching stopped by a dummy access to S1BIT. In the receive loop both BB and RBF will be checked. When BB is cleared, a STOP condition is detected and the routine will be left with CY=0.
The first 7 bits are received via S1BIT because this will release stretching. The 8th bit is accessed via SDI because stretching must be maintained.

If the slave routine is left via EXIT, the STR bit is cleared (to disable stretching on SCL edges when the 8xC528 is not addressed as slave) and a dummy access to S1BIT is done to finish current SCL stretching. If the slave routine was entered via an interrupt the previous context is restored.

```
TSW-ASM51 V3.0b Serial #00052252  Slave interrupt routine
PAGE   1

LOC   OBJ          LINE  SOURCE

                      1  $TITLE(Slave interrupt routine)
                      2  $DEBUG
                      3  $NOLIST
                      6  ;
                      7  ;This routine handles I2C interrupts.
                      8  ;8xC528 I2C interface enters in slave mode.
                      9  ;After testing R/W bit, 8xC528 will go in slave-transmit or
                     10  ;slave-receive mode.
                     11  ;Source or destination buffer for data uses pointer SLAVE_SUB_ADDRESS
                     12  ;Slave routine will use register bank 01
                     13  ;
                     14  ;*******************************************************************
                     15  ;Interrupt entry point
                     16
----                 17                  CSEG AT 53H
                     18
C053: 020000    R    19                  LJMP __PIP0A    ;Vector to interrupt handler
                     20  ;
                     21  ;*******************************************************************
                     22
                     23                  I2C_DRIVER SEGMENT CODE INBLOCK
----                 24                  RSEG I2C_DRIVER
                     25
                     26                  PUBLIC MST_ENTRY
                     27                  EXTRN  DATA(SLAVE_SUB_ADDRESS)
                     28                  EXTRN  BIT(ARB_LOST)
                     29
REG END              30                  BUF_POINT SET R0
REG END              31                  OWN_SLAVE SET R7
REG END              32                  BIT_CNT   SET R2
                     33
                     34 ;*******************************************************************
                     35
0000: C0E0      R    36  __PIP0A:PUSH ACC        ;Push CPU status on stack
0002: C0D0           37          PUSH PSW
0004: 75D008         38          MOV PSW,#08H    ;Select registerbank 01
                     39
                     40 ;*******************************************************************
                     41  ;Check slave address
                     42 ;*******************************************************************
                     43
0007: 43D842         44          ORL S1SCS,#01000010B ;Disable SCL generation and enable SCL
                                                      ;stretching stretching
000A: 1142      R    45          ACALL RCV_SL_BY ;Receive slave address, on exit SCL is
                                                 ; stretched
000C: A2E0           46  PROC:   MOV C,ACC.0     ;Store R/W bit in F0
000E: 92D5           47          MOV F0,C
0010: 6F             48          XRL A,OWN_SLAVE ;Compare received slave address
```

```
TSW-ASM51 V3.0b Serial #00052252  Slave interrupt routine
PAGE   2

LOC   OBJ          LINE  SOURCE

0011: C2E0           49          CLR ACC.0       ;Ignore R/W bit
0013: 7050           50          JNZ NO_MATCH    ;Leave slave-routine if there is no match
0015: C3             51          CLR C           ;Send ACK
0016: 115C     R     52          ACALL SEND_ACK
0018: A800     R     53          MOV BUF_POINT,SLAVE_SUB_ADDRESS  ;Get buffer pointer
001A: A2D5           54          MOV C,F0        ;Restore R/W bit
001C: 5019           55          JNC RCV_SLAVE   ;Test R/W bit
                     56
                     57
                     58  ;*********************************************************************
                     59  ;Slave transmitter mode
                     60  ;*********************************************************************
                     61
                     62
001E: E6             63  NXT_TRX:MOV A,@BUF_POINT;Get byte to send
001F: 7A08           64          MOV BIT_CNT,#08 ;Init bit counter
                     65
0021:                66  NXT_TRX_BIT:
0021: F5D9           67          MOV S1BIT,A     ;Trx bit and stretch after transmission
0023: 23             68          RL A            ;Prepare next bit to send
0024: 30DAFD         69          JNB WBF,$       ;Test if bit is sent
0027: DAF8           70          DJNZ BIT_CNT,NXT_TRX_BIT ;Test if all bits are sent
                     71
0029: D2DF           72          SETB SDO        ;Release SDA line for NO_ACK/ACK reception
002B: E5D9           73          MOV A,S1BIT     ;Stop stretching
002D: 30DBFD         74          JNB RBF,$       ;Test is ACK bit is received
0030: A2DF           75          MOV C,SDI       ;Read bit, SCL remains stretched
0032: 4040           76          JC EXIT         ;NO_ACK received. Exit slave routine
0034: 08             77          INC BUF_POINT   ;ACK received. Update pointer for next byte to
                                                 ;send
0035: 80E7           78          SJMP NXT_TRX
                     79
                     80  ;*********************************************************************
                     81  ;Slave receiver mode
                     82  ;*********************************************************************
                     83
0037:                84  RCV_SLAVE:             ;Entry in slave-receiver mode
0037: 1142     R     85          ACALL RCV_SL_BY ;Receive byte
0039: 5039           86          JNC EXIT        ;If STOP is detected, then exit
003B: F6             87          MOV @BUF_POINT,A;Store received byte
003C: C3             88          CLR C           ;Send ACK
003D: 115C     R     89          CALL SEND_ACK
003F: 08             90          INC BUF_POINT   ;Update pointer
0040: 80F5           91          SJMP RCV_SLAVE  ;Receive next byte
                     92
                     93
```

```
TSW-ASM51 V3.0b Serial #00052252  Slave interrupt routine
PAGE   3

LOC   OBJ          LINE  SOURCE

                     94  ;*********************************************************************
                     95  ;Receive byte routine
                     96  ;On exit, received byte in accu
                     97  ;On exit CY=0 if STOP is detected
                     98  ;*********************************************************************
0042:                99  RCV_SL_BY:
0042: 7A08          100          MOV BIT_CNT,#08
0044: E5D9          101          MOV A,S1BIT      ;Disable stretching from START or previous ACK
0046: E4            102          CLR A
0047:               103  RCV_BIT:
0047: 30DC10        104          JNB BB,STOP_RCV ;Test if STOP-condition is received
004A: 30DBFA        105          JNB RBF,RCV_BIT ;Wait till received bit is valid
004D: BA0105        106          CJNE BIT_CNT,#01,ASSEM_BIT ;Check if last bit is to be received
                    107
0050: A2DF          108          MOV C,SDI        ;Get last bit without stopping stretching
0052: 33            109          RLC A
0053: D3            110          SETB C           ;No STOP detected
0054: 22            111          RET
                    112
0055:               113  ASSEM_BIT:
0055: 45D9          114          ORL A,S1BIT      ;Receive bit; release RBF,CLH and SCL stretching

0057: 23            115          RL A
0058: DAED          116          DJNZ BIT_CNT,RCV_BIT
                    117
005A:               118  STOP_RCV:
005A: C3            119          CLR C            ;STOP detected
005B: 22            120          RET
                    121
                    122 ;*********************************************************************
                    123  ;Send ACK/NO_ACK. Value of ACK in Carry
                    124 ;*********************************************************************
005C:               125  SEND_ACK:
005C: 13            126          RRC A
005D: F5D9          127          MOV S1BIT,A      ;Carry to SDA line
005F: 30DAFD        128          JNB WBF,$        ;Test if ACK/NO_ACK is sent
0062: D2DF          129          SETB SDO         ;Release SDA line
0064: 22            130          RET
                    131
                    132 ;*********************************************************************
                    133  ;No match between received slave-address and own-slave-address
                    134 ;*********************************************************************
0065:               135  NO_MATCH:
0065: D3            136          SETB C           ;Send NO_ACK
0066: 115C      R   137          ACALL SEND_ACK
0068: 800A          138          SJMP EXIT
```

```
TSN-ASM51 V3.0b Serial #00052252  Slave interrupt routine
PAGE   4

                    139  ;*********************************************************************
                    141  ;Entry point when an arbitration-lost condition is detected in
                         ;master-mode.
                    142  ;*********************************************************************
006A:               143  MST_ENTRY:
006A: 23            144          RL A            ;Restore slave address  sofar
006B: C2E0          145          CLR ACC.0
006D: 43D842        146          ORL S1SCS,#01000010B ;Disable SCL generation and enable SCL
                                                  ;stretching
0070: 1147      R   147          ACALL RCV_BIT   ;Proceed with receiving rest of slave address
0072: 8098          148          SJMP PROC
                    149
                    150
                    151 ;*********************************************************************
                    152  ;Exit from interrupt routine
                    153 ;*********************************************************************
                    154
0074: C2D9          155  EXIT:   CLR STR         ;Disable stretching on next falling SCL edges
0076: E5D9          156          MOV A,S1BIT     ;Stop current SCL stretching
0078: 300001    R   157          JNB ARB_LOST,EX_SL
007B: 22            158          RET             ;Exit when entered from master mode
007C: D0D0          159  EX_SL:  POP PSW         ;Restore old CPU status
007E: D0E0          160          POP ACC
0080: 32            161          RETI
                    162
0081:               163          END
```

4. EXAMPLES

4.1 INTRODUCTION

Some examples are given how to use the I²C routines in an application program. Examples are given for an assembly, PL/M and C program.
The program displays time from the PCF8583P clock/calendar/RAM on an LCD display driven by the PCF8577.
The example can be executed on the OM4151 I²C evaluation board.

4.2 Using the routines with assembly sources

From page 35 the listing is shown of the example program. The most important aspect when using the I²C routines, is preparing the input parameters before the sub-routine call.
When for example the IIC_Write routine must be called, the parameters must be called in the following order:

```
MOV _IIC_READ_BYTE,#SLAVE_ADR
MOV _IIC_READ_BYTE+1,#COUNT_1
MOV _IIC_READ_BYTE+2,#SOURCE_PTR_1
CALL _IIC_READ
```

Note that the order of defining the parameters is the same as in a PL/M-call (see page 22).
An easier way to call the routines is making a macro that includes the initialising of the parameters.
The example program makes use of macros.
IIC_Read is then called in the following way:

```
%IIC_Read(Slave_Adr,Count_1,Source_Ptr_1);
```

Note that in the listing the contents of the macro are shown, in stead of the call.

The macro must be written as follows:

```
%* DEFINE (IIC_Read(SLAVE_ADR,COUNT_1,SOURCE_PTR_1))(
   MOV _IIC_READ_BYTE,#%SLAVE_ADR
   MOV _IIC_READ_BYTE+1,#%COUNT_1
   MOV _IIC_READ_BYTE+2,#%SOURCE_PTR_1
   LCALL _IIC_READ)
```

Macro's for the I²C CALL's are found in I2C.MAC. This file should be included in all modules making use of the macro's. One of the modules should also include the variable definitions needed by the I²C routines. These are found in file VAR_DEF.ASM. If the program consists of more than 1 module, then these modules should also include EXT_VAR.ASM. This file contains the EXTRN-definitions of the I²C routines.

When an I²C routine is called the accumulator contains status information and the CY-bit is set if an error has occurred. The contents of the accumulator are the same as the returned byte when using PL/M (see pg. 10).

I²C routines for 8XC528 AN438

```
TSW-ASM51 V3.0b Serial #00052252  Assembly example program                    PAGE   1

LOC   OBJ          LINE  SOURCE

                      1  $TITLE(Assembly example program)
                      2  $DEBUG
                      3
                      4  ;Hours and minutes will be displayed on LCD display
                      5  ;Dot between hours and minutes will blink
                      6
                      7  $                                        ;Include I2C var. definitions
                      8  # 1 "C:\USER\VAR_DEF.ASM"
                     74  # 8 "DEMO_ASM.ASM"
                      8  $                                        ;Include I2C macro's
                      9  # 1 "C:\USER\I2C.MAC"
                     35  # 9 "DEMO_ASM.ASM"
00A2                 10  CLOCK_ADR   EQU 0A2h                     ;Address of PCF8583
0001                 11  CL_SUB_ADR  EQU 01h                      ;Sub address for reading time
0074                 12  LCD_ADR     EQU 74h                      ;Address of PCF8577
                     13
                     14  RAMVAR  SEGMENT DATA                     ;Segment for variables
                     15  USER    SEGMENT CODE                     ;Segment for application
                                                                  ;program
                     16
----                 17          RSEG RAMVAR
0000:           R    18  STACK:      DS 10                        ;Stack area
000A:                19  TIME_BUFFER:DS 4                         ;Buffer for I2C strings
000E:                20  LCD_BUFFER: DS 5
                     21
----                 22          CSEG AT 00
0000: 020000    R    23          LJMP APL_START
                     24
                     25          .
----                 26          RSEG USER
                     27
0000:           R    28  APL_START:
0000: 900073    R    29          MOV DPTR,#LCD_TAB                ;Pointer to segment table
0003: 7581FF    R    30          MOV SP,#STACK-1                  ;Initialise stack
0006: 750E00    R    31          MOV LCD_BUFFER,#00               ;Control word for LCD driver
                     32
0009: 750022    R    33      MOV _Init_IIC_Byte  ,#22h
000C: 75010A    R    34      MOV _Init_IIC_Byte+1,#TIME_BUFFER
000F: 120000    R    35      LCALL _Init_IIC
                     36            ;Initialise I2C interface
0012: E4             37          CLR A                            ;Prepare buffer for clock int.
0013: F50A      R    38          MOV TIME_BUFFER,A
0015: F50B      R    39          MOV TIME_BUFFER+1,A
                     40
0017: 7500A2    R    41      MOV _IIC_Write_Byte  ,#CLOCK_ADR
001A: 750102    R    42      MOV _IIC_Write_Byte+1,#2
001D: 75020A    R    43      MOV _IIC_Write_Byte+2,#TIME_BUFFER
0020: 120000    R    44      LCALL _IIC_Write
```

```
TSW-ASM51 V3.0b Serial #00052252  Assembly example program                    PAGE   2

LOC   OBJ          LINE  SOURCE

                     45       ;Initialise clock
                     46
0023:                47  REPEAT:
0023: 7500A2    R    48       MOV _IIC_Read_Sub_Byte  ,#CLOCK_ADR
0026: 750104    R    49       MOV _IIC_Read_Sub_Byte+1,#4
0029: 75020A    R    50       MOV _IIC_Read_Sub_Byte+2,#TIME_BUFFER
002C: 750301    R    51       MOV _IIC_Read_Sub_Byte+3,#CL_SUB_ADR
002F: 120000    R    52       LCALL _IIC_Read_Sub
                     53        ;Read time
                     54
                     55  ;Time has been read. Order: hundreds of sec's, sec's, min's and hr's
0032: E50D      R    56           MOV A,TIME_BUFFER+3             ;Mask of hour counter
0034: 543F           57           ANL A,#3Fh
0036: F50D      R    58           MOV TIME_BUFFER+3,A
                     59
0038: 120054    R    60           CALL CONVERT                    ;Convert time data to LCD
                                                                  ;segment data
                     61
                     62  ;Check if dot has to be switched on
003B: 431101    R    63           ORL LCD_BUFFER+3,#01h
                     64  ;If lsb of seconds is '0', then switch on dp
003E: E50B      R    65           MOV A,TIME_BUFFER+1             ;Get seconds
0040: 13             66           RRC A
0041: 4003           67           JC PROCEED
0043: 430F01    R    68           ORL LCD_BUFFER+1,#01            ;Switch on dp
                     69
                     70  ;Display new time
0046:                71  PROCEED:
0046: 750074    R    72       MOV _IIC_Write_Byte  ,#LCD_ADR
0049: 750105    R    73       MOV _IIC_Write_Byte+1,#5
004C: 75020E    R    74       MOV _IIC_Write_Byte+2,#LCD_BUFFER
004F: 120000    R    75       LCALL _IIC_Write
                     76
0052: 80CF           77           SJMP REPEAT                     ;Read new time
                     78
                     79
                     80  ;CONVERT converts BCD data of time to segment data
0054: 780F      R    81  CONVERT:MOV R0,#LCD_BUFFER+1             ;R0 is pointer
0056: E50D      R    82           MOV A,TIME_BUFFER+3             ;Get hours
0058: C4             83           SWAP A                          ;Swap nibbles
0059: 12006D    R    84           CALL LCD_DATA                   ;Convert 10's of hours
005C: E50D      R    85           MOV A,TIME_BUFFER+3
005E: 12006D    R    86           CALL LCD_DATA                   ;Convert hours
0061: E50C      R    87           MOV A,TIME_BUFFER+2             ;Get minutes
0063: C4             88           SWAP A
0064: 12006D    R    89           CALL LCD_DATA                   ;Convert 10's of minutes
0067: E50C      R    90           MOV A,TIME_BUFFER+2
0069: 12006D    R    91           CALL LCD_DATA                   ;Convert minutes
```

```
TSW-ASM51 V3.0b Serial #00052252  Assembly example program                    PAGE   3

LOC   OBJ         LINE  SOURCE

006C: 22            92          RET
                    93
                    94  ;LCD_DATA gets data from segment table and stores it in LCD_BUFFER
006D:               95  LCD_DATA:
006D: 540F          96          ANL A,#0FH                      ;Mask off LS-nibble
006F: 93            97          MOVC A,@A+DPTR                  ;Get segment data
0070: F6            98          MOV @R0,A                       ;Save segment data
0071: 08            99          INC R0
0072: 22           100          RET
                   101  ;
                   102  ;Conversion table for LCD
0073:              103  LCD_TAB:
0073: FC60DA       104          DB 0FCH,60H,0DAH                ;'0','1','2'
0076: F266B6       105          DB 0F2H,66H,0B6H                ;'3','4','5'
0079: 3EE0FE       106          DB 3EH,0E0H,0FEH                ;'6','7','8'
007C: E6           107          DB 0E6H                         ;'9'
                   108  ;
007D:              109          END
```

4.3 Using the routines with PL/M-51 sources

The listing from pg. 39 shows the listing of the clock program in PL/M-51. The procedures are untyped. The routines are used the same way as in the examples of chapter 3.2

```
$OPTIMIZE(4)
$DEBUG
$CODE

/* Hours and minutes will be displayed on LCD display
   Dots between hours and minutes will blink */

Demo_plm: Do;

/* External declarations */

Init_IIC: Procedure(Own_Adr,Slave_Ptr) External;
          Declare (Own_Adr,Slave_Ptr) Byte Main;
End Init_IIC;

IIC_Write: Procedure(Sl_Adr,Nr_Bytes,Source_Ptr) External;
           Declare (Sl_Adr,Nr_Bytes,Source_Ptr) Byte Main;
End IIC_Write;

IIC_Read_Sub: Procedure(Sl_Adr,Nr_Bytes,Dest_Ptr,Sub_Adr) External;
              Declare(Sl_Adr,Nr_Bytes,Dest_Ptr,Sub_Adr) Byte Main;
End IIC_Read_Sub;
```

```
Clock: Do;
         /* Variable and constant declarations */

         Declare LCD_TAB(*) Byte Constant (0FCh,60H,0DAH,0F2H,66H,
                                           0B6H,3EH,0E0H,0FEH,0E6H);
         Declare Time_Buffer(4) Byte Main;
         Declare LCD_Buffer(5) Byte Main;
         Declare Tab_Point Word Main;
         Declare (LCD_Point,Time_Point) Byte Main;
         Declare Segment Based LCD_Point Byte Main;
         Declare Time Based Time_Point Byte Main;
         Declare Tab_Value Based Tab_Point Byte Constant;

         Declare Clock_Adr Literally '0A2h';
         Declare LCD_Adr Literally '74h';
         Declare Cl_Sub_Adr Literally '01h';

         Call Init_IIC(22h,.Time_Buffer);
         LCD_Buffer(0)=0; /* LCD control word */
         Time_Buffer(0)=0;
         Time_Buffer(1)=0;
         Call IIC_Write(Clock_Adr,2,.Time_Buffer); /* Initialise clock */

         Do While LCD_Buffer(0)=0; /* Program loop */
            Call IIC_Read_Sub(Clock_Adr,4,.Time_Buffer,Cl_Sub_Adr);
                                                  /* Get time */
            LCD_Point=.LCD_Buffer+1; /* Initialise pointers */
            Time_point=.Time_Buffer(3);
            Tab_Point=.LCD_Tab(0)+SHR(Time,4); /* 10-HR's */
            Segment=Tab_Value;
            LCD_Point=LCD_Point+1;
            Tab_Point=.LCD_Tab(0)+(Time AND 0FH); /* HR's */
            Segment=Tab_Value;
            Time_Point=Time_Point-1;
            LCD_Point=LCD_Point+1;
            Tab_Point=.LCD_Tab+SHR(Time,4); /* 10-MIN's */
            Segment=(Tab_Value OR 01H); /* dp */
            LCD_Point=LCD_Point+1;
            Tab_Point=.LCD_Tab+(Time AND 0FH); /* MIN's */
            Segment=Tab_Value;
            Time_Point=.Time_Buffer(1)+1; /* Check sec's for blinking */
            LCD_Point=.LCD_Buffer+1;
            If (Time AND 01H)>0 then Segment=(Segment OR 01H);
            Call IIC_Write(LCD_Adr,5,.LCD_Buffer); /* Display time */
         End;

      End Clock;

End Demo_plm;
```

4.4 Using the routines with C sources

On page 42 an example of a C program is shown using the I2C routines. Function prototypes are found in header file "i2c.h". In this example the function prototypes are written in such a way that no value is returned by the function. If the STATUS byte is needed, the header file may be changed to return a byte.
Note that the function calls are written in upper-case. This is due to the fact that the used version of the assembler/linker is case sensitive.

```
#include <C:\USER\i2c.h>

        rom  char              LCD_Tab[]={0xFC,0x60,0xDA,0xF2,0x66,0xB6,0x3E,
                                          0xE0,0xFE,0xE6};

void main()

{

       #define Clock_Adr       0xA2
       #define LCD_Adr         0x74
       #define Cl_Sub_Adr      0x01

       rom  char               * Tab_Ptr;
       data char               Time_Buffer[4];
       data char               * Time_Ptr;
       data char               LCD_Buffer[5];
       data char               * LCD_Ptr;

       INIT_IIC(0x22,&Time_Buffer);
       LCD_Buffer[0]=0; /* LCD control word */
       Time_Buffer[0]=0;
       Time_Buffer[1]=0;
       IIC_WRITE(Clock_Adr,2,&Time_Buffer); /* Initialise clock */

       while (1)       /* Program loop */
             {
              IIC_READ_SUB(Clock_Adr,4,&Time_Buffer,Cl_Sub_Adr);
                                                  /* Get time */
              LCD_Ptr = &LCD_Buffer[1]; /* Initialise pointers */
              Time_Ptr = &Time_Buffer[3];
              Tab_Ptr = (LCD_Tab+(*Time_Ptr >> 4)); /* 10-HR's */
              *(LCD_Ptr++) = *Tab_Ptr;
              Tab_Ptr = (LCD_Tab+(*(Time_Ptr--) & 0x0F)); /* HR's */
              *(LCD_Ptr++) = *Tab_Ptr;
              Tab_Ptr = (LCD_Tab+(*Time_Ptr >> 4)); /* 10-MIN's */
              *(LCD_Ptr++) = (*Tab_Ptr | 0x01); /* dp */
              Tab_Ptr = (LCD_Tab+(*Time_Ptr & 0x0F)); /* MIN's */
              *LCD_Ptr = *Tab_Ptr;
              Time_Ptr = &Time_Buffer[1]; /* Check sec's for blinking */
              LCD_Ptr = &LCD_Buffer[1];
              if ((*Time_Ptr & 0x01)>0)
                 *LCD_Ptr = (*LCD_Ptr | 0x01);
              IIC_WRITE(LCD_Adr,5,&LCD_Buffer); /* Display time */
             }
}
```

5: CONTENTS OF DISK

A disk contains the following 3 directories:

1: \USER
This directory contains the files that may be used in the user program.

I2C_DR.LIB	Library with I²C routines.
I2C.H	Header file for C applications.
I2C.MAC	Macro's for the I²C routine calls in assembly programs.
VAR_DEF.ASM	Include file with variable definitions for assembly programs.
EXT_VAR.ASM	Include file with external definitions for assembly programs.
LIB.BAT	Example batch file to create I2C_DR.LIB.
ASM.BAT	Examplebatch file to assemble source modules for library.

2: \EXAMPLE
This directory contains the source files of the examples described in chapter 4.

DEMO_ASM.*	Assembly example.
DEMO_PLM.*	PL/M example.
HEAD_51.SRC	Example of environment file for PL/M example.
DEMO_C.*	C example.
CSTART.ASM	Example of environment file for C example.

3: \SOURCE
This directory contains the source files of the modules in the library.

Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specifications defined by Philips.

(BCM) 87C751 Specification for a bus-controlled monitor

AN442

SUMMARY

BCM87C751 is a powerful, flexible and low cost Digital Controlled Monitor System, based on the 87C751 microcontroller. It employs I²C bus control with various I²C bus controlled peripherals (PCF8582EP–EEPROM and TDA8444 D/A conterter). The control function is implemented via 8 6-bit DC voltage output from TDA8444.

Some features of the system:

- Flexible approach, especially for multisync or auto sync operation
- Mode detection and frequency measurements by microprocessor
- Mode switching under software control
- Elimination of potentiometers
- Quick factory alignment (DACs can be preset)
- Automatic factory alignment possible

This document describes the operation and the use of the system. it provides necessary information concerning operation, required hardware, flow charts and their effect on the performance.

INTRODUCTION

Figure 1 shows the block diagram of a high-performance color monitor with microcontroller and several parts that communicate via the two-wire I²C-bus.

The system can perform the following:

- Determine the mode and standard of incoming signals with the stored values in memory (e.g. multisync modes).
- Enter parameters of user defined modes into memory via a keyboard.
- Control analog parameters such as contrast and brightness via the bus from keyboard inputs.
- Control mode and standard parameters (such as picture geometry parameters and the free-running oscilltor frequency).

Features

- Multisync or Autosync operation
  - FH = 31kHz–95kHz
  - FV = 50Hz–114Hz
- Selectable four or five function control configurations
- Selectable one digit, or one and one-half digit, or null seven segment mode display.
- Selectable Horizontal dirtect ratio or indirect ratio F to V converter DAC output configurations
- Selectable ten user modes plus ten factory modes, or, one user mode plus 19 factory modes configurations
- Selectable multiple up/down keys or minimum keyboard configuration
- Change function without save key, all keys but the reload key have a repeat function
- Both Horizontal and Vertical outputs have F-to-V converter DAC outputs
- Four outputs of Horizontal PLL capacitor selection signal. This can be easily adapted for further extension.

IC DESCRIPTIONS

87C751

A derivative of the 8051 family of microcontrollers, the 87C751 has an 8-bit CPU, 2K bytes EPROM, 64 bytes RAM, 19 I/O lines, a bi-directional inter-integrated circuit (I²C) serial bus interface, and an on-chip oscillator.

PCF8581/2

A 1K- or 2K-bit, 5V electrically erasable programmable read only memory (EEPROM) organized as 128 or 256 × 8 bits. The stored information is electrically alterable on a word-by-word basis, I²C-bus controlled.

TDA8444

Comprises eight DACs, each controlled via the I²C-bus. The DACs are individually programmed using a 6-bit word to select an output from one of 64 voltage steps. the maximum output voltage of all DACs is set by the input VMAX and the resolution is approximately VMAX/64.

For detailed information on these devices, please refer to their respective data sheets.

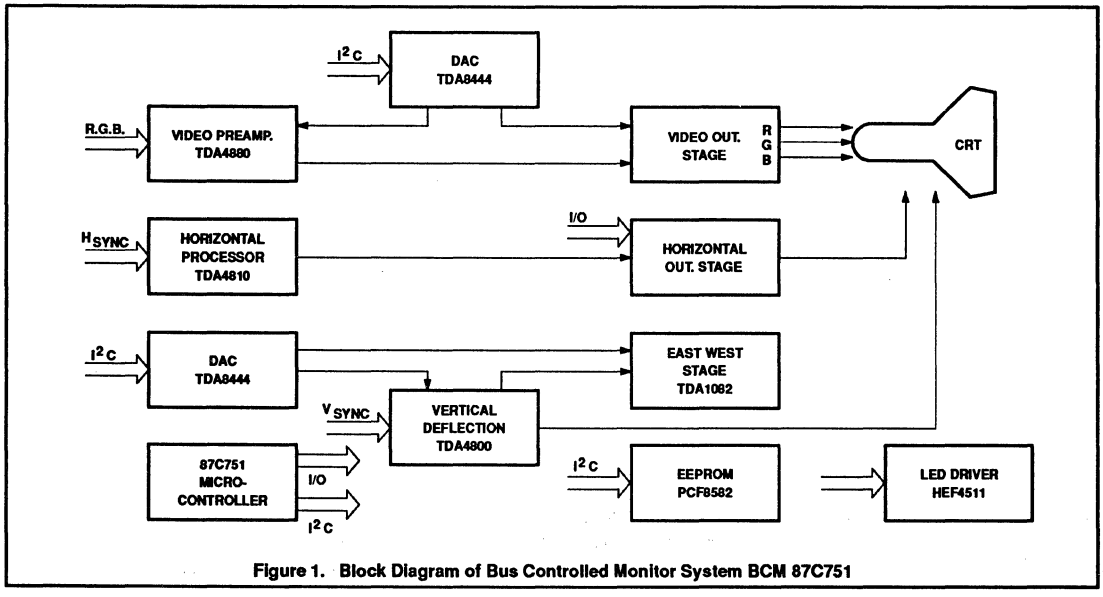


Figure 1. Block Diagram of Bus Controlled Monitor System BCM 87C751

(BCM) 87C751 Specification for a bus-controlled monitor

AN442

OPERATION INSTRUCTION

Function Selection and Change

When the monitor is powered on it will automatically enter the corresponding mode depending on the input signal. Different configurations and functions can be defined by users with the use of different features. The following configuration's functions are examples for user's reference. See Figure 2 for Configurations 1 through 4.

To adjust functions such as V-size, V-shift, H-center, H-shift, and PCC (Pincushion Correction Circuitry):

1. The "Function" key should be pressed until the required function LED is lit.
2. If the V-shift LED is on, the user can then adjust V-shift by pressing Up or Down key. If the Up key is pressed, the V-shift DAC output will increase one step. While the Down key is pressed, the V-shift DAC output will decrease one step. The user can repeat the Up or Down key simply by pressing it longer than 0.5 second. It will then automatically repeat approximately 2 times per second until the key is released.

Mode Selection and Change

(See Figure 4)

In Figures 3 and 4. the mode number is displayed on the two seven-segment LEDs. Each number denotes one mode. Modes 10 through 19 are user modes, which can be defined by the user. When there is a new mode entering the monitor that does not belong to any mode stored n the EEPROM, the mode display will show "19". If the user presses the Reload key while the mode display is "19", the display will flash. When the mode display is flashing, the user can select the destination mode by pressing the Up/Down keys. The destination mode is between 10 and 19.

Every press of the Up key causes the flashing display to add one, unless it already reached 19. Every press of the Down key causes the flashing display to subtract one, unless is has reached 10. When the user lets the destination mode flash on the display, the user can press the Relead key to store the new mode to destination mode. When the mode display stops flashing, the new mode is stored. The newly stored destination mode is permanent, unless the user repeats the entire procedure.

To change an old user mode, already stored in EEPROM, to a different user mode, press the Reload key for longer than 8 seconds while the monitor is working in the old mode. The mode display will flash the old mode, then the user can use the Up/Down key to select the new mode. Press the Reload key again to copy the old to new destination user mode. If the user forgets to press the Reload key again, the flashing of the mode display will last for 2 minutes, then the program will cancel the copy old mode to new user mode command.

During the flash period, the program still monitors the Horizontal and Vertical sync signal to adapt the DAC to the proper mode. For example, while the user tries to copy new mode 13 to mode 15, and the mode display 13 (15) is flashing, if the PC sends out a signal for mode 3, the program will change the DAC output to adjust the monitor to work in mode 3, but the mode display is still flashing on user mode 13 (15) and the store procedure is still going on until the user presses the Reload key again, or terminates the copy procedure after the 2 minutes time out. The mode display then shows 3.

NOTE: Upon request, we can also program in advance those 10 user-defined modes that can still be changed by the user if necessary for further extension.

For Configuration 5 to Configuration 8, see Figure 3.

To adjust functions, the user can simply press the corresponding push button. The upper push button will increase the DAC output. The lower push button will decrease the DAC output. (A total of 64 steps can be programmed in advance.)

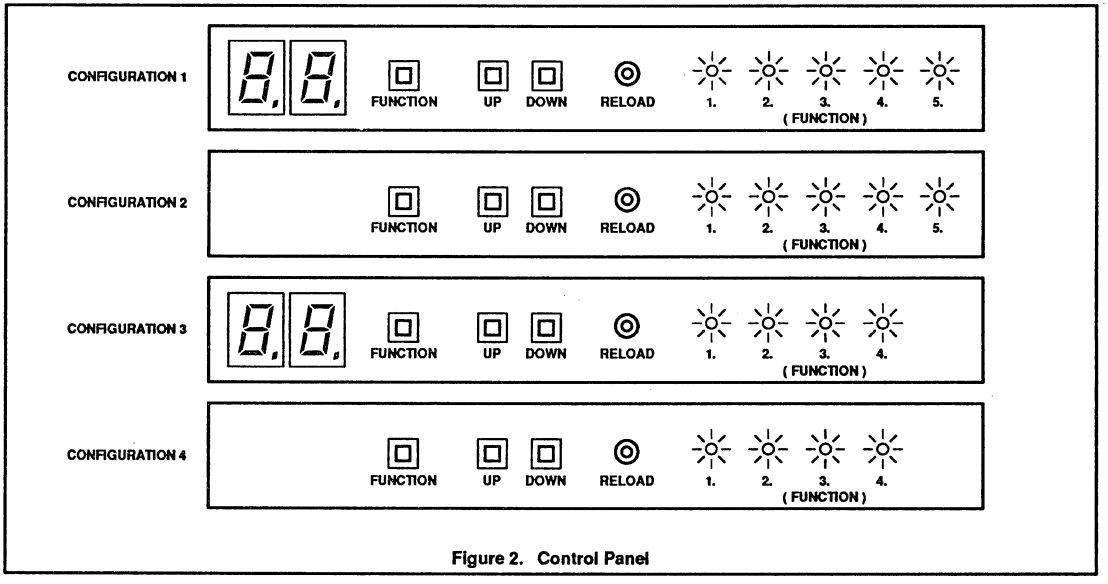


Figure 2. Control Panel

(BCM) 87C751
Specification for a bus-controlled monitor

AN442

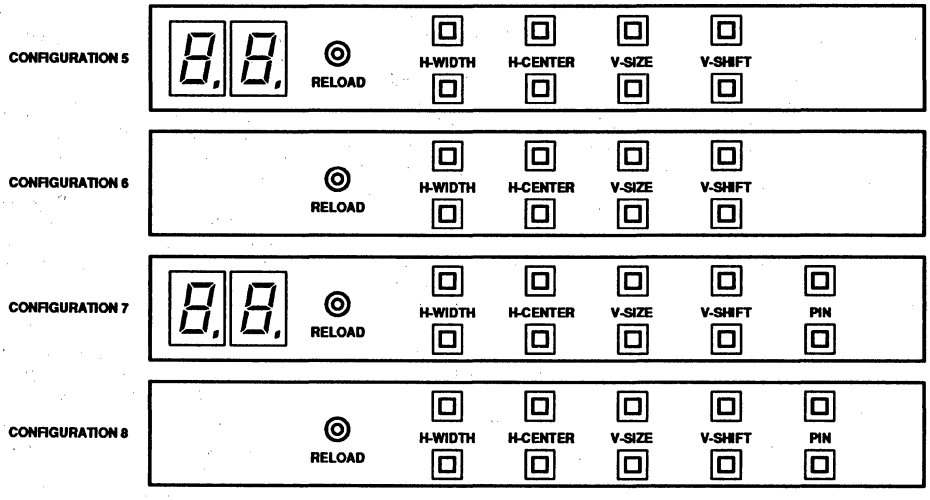


Figure 3. Control Panel

MODE	NAME	H.F.	V.F.	H.P.	V.P.
0	VGA-1	31K	70	+	–
1	VGA-2	31K	70	–	+
2	VGA-3	31K	60	–	–
3	8514A	35K	87	+	+
4	SVGA-1	35K	56	+	+
5	UVGA-1	48K	60	+	+
6	VESA	56K	70	–	–
7	V64-1	64K	60	+	+
8	SVGA-2	37K	60	+	+
9	V78	78K	60	+	+
10	USER DEFINE	·	·	·	·
·	·	·	·	·	·
18	·	·	·	·	·
19	·	·	·	·	·

Figure 4. Mode Selection

(BCM) 87C751 Specification for a bus-controlled monitor

AN442

SOFTWARE FLOW CHART DESCRITPION

(See Figures 5 through 7)

When power is on, software intializes the hardware first. The microcontroller waits 100μs for the settlement of the hardware, then initializes itself by specifying stack, setting timer, clearing RAM, arranging interrupt, . . . , etc.

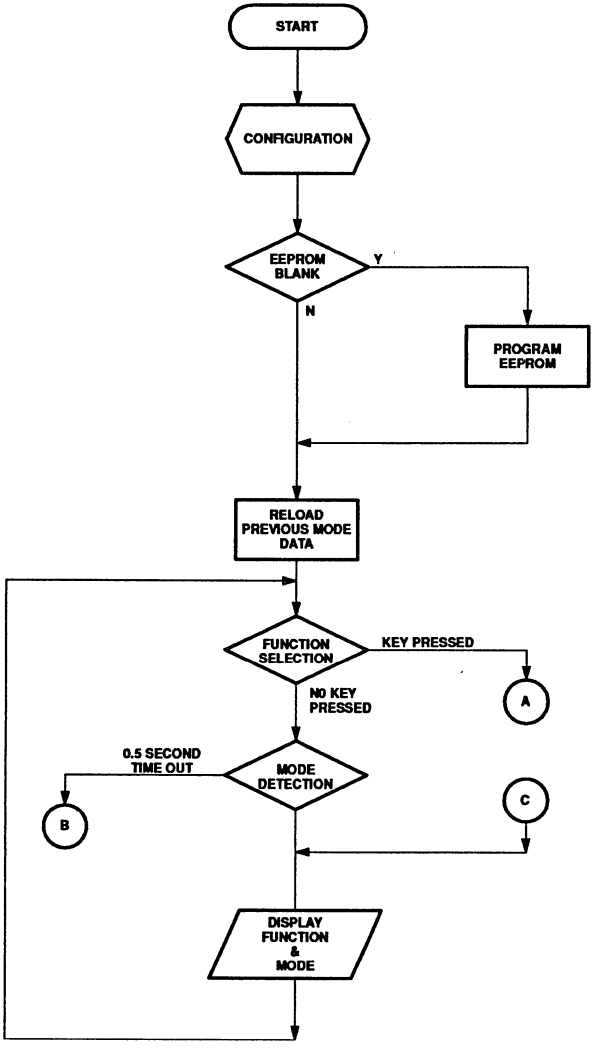


Figure 5. Main Flow Chart

(BCM) 87C751 Specification for a bus-controlled monitor

AN442

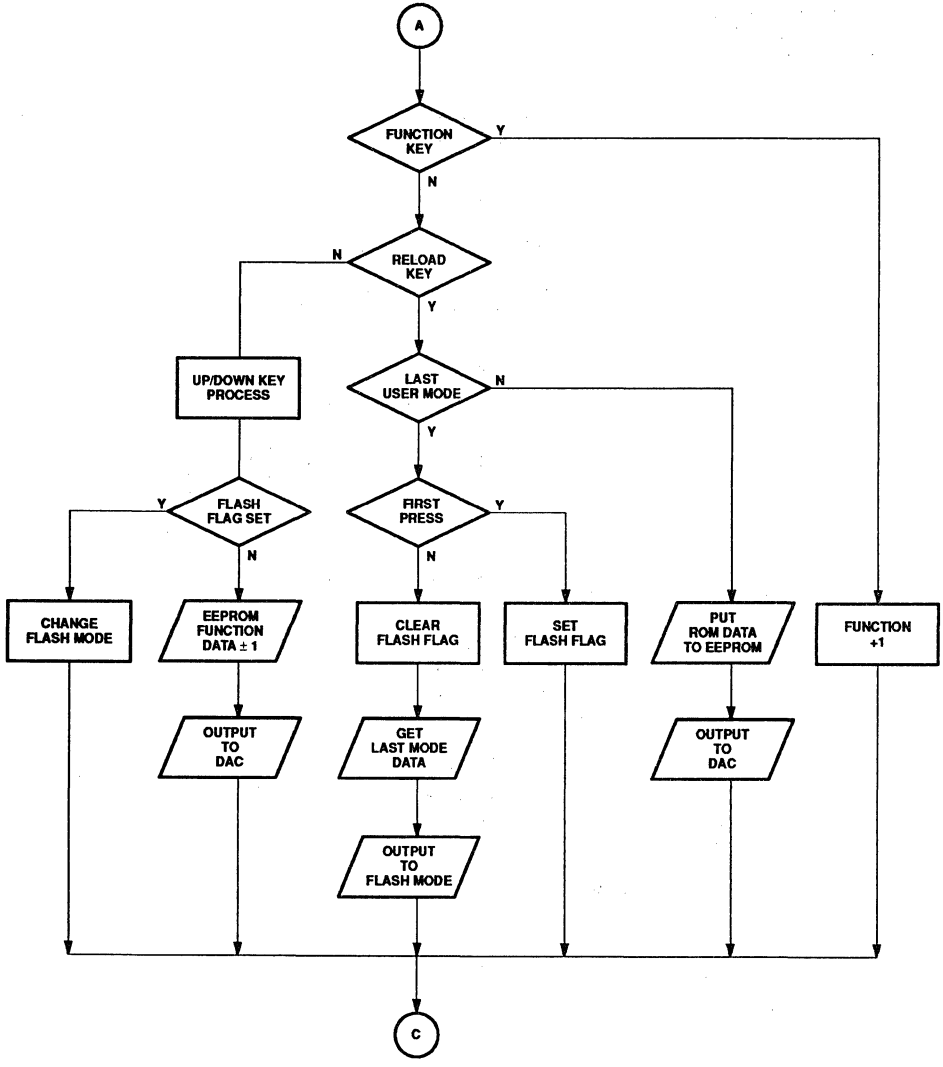


Figure 6. Function Selection Flow Chart

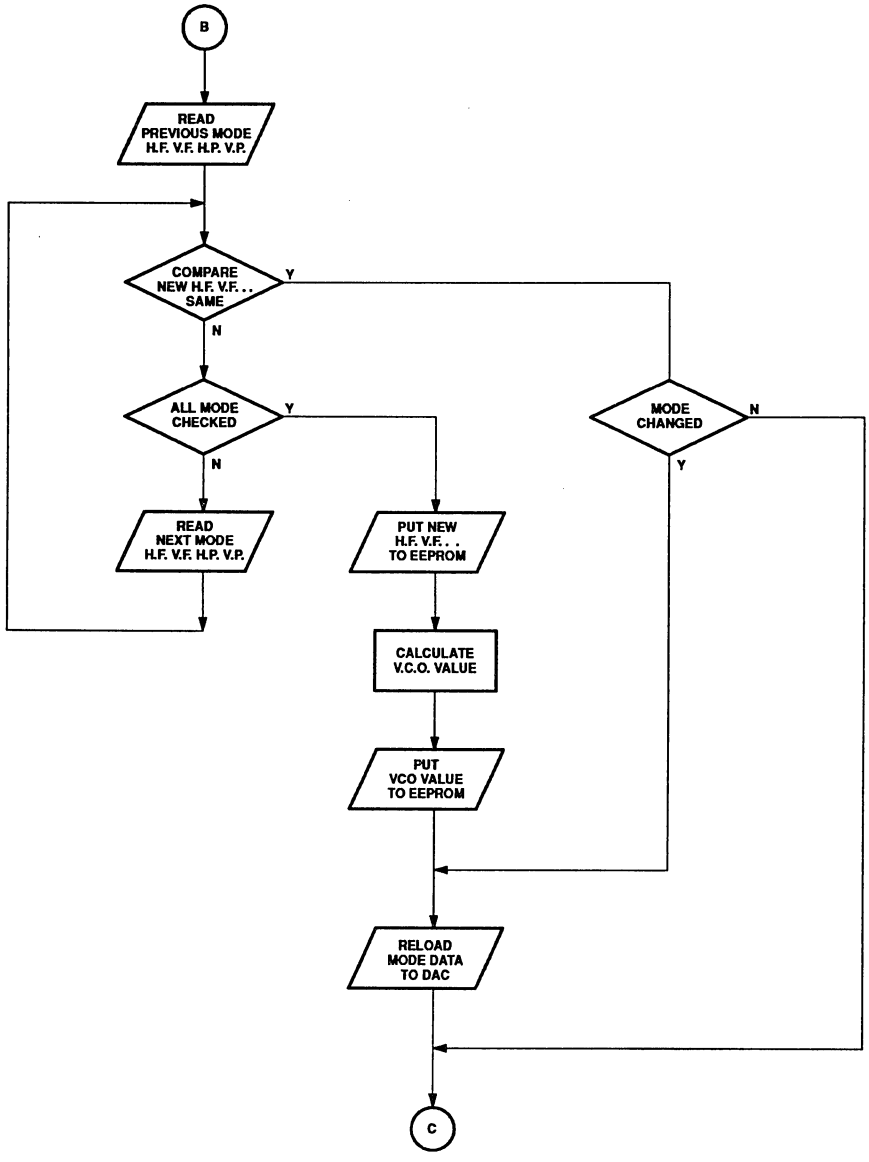


Figure 7. Mode Detection Flow Chart

(BCM) 87C751 Specification for a bus-controlled monitor

Table 1.

	ADD DIODE	REMOVE DIODE
D1	Indirect ratio of VCO output (JP4)—See Figure 8.	Direct ratio of VCO output (JP4)—See Figure 8.
D2	10 user modes 10 factory modes (JP5)	1 user mode 19 factory modes (JP5)
D3	4 adjustable functions (JP6)	5 adjustable functions (JP6)
D4	Multiple Up/Down keys (JP7)	Single Up/Down keys (JP7)

Referring to previous section, *Function Selection and Change*, software can detect the hardware configuration by pulling 87C751 microcontroller pin P0.2 LOW to read the diode arrangement. Each diode denotes one change in the hardware configuration. Table 1 explains the usage of each diode.

After the configuration detection, then it goes to check EEPROM status. If the EEPROM is blank, the program will start to move all factory set data from the microcontroller's PROM to EEPROM. The last byte datum in the microcontroller's PROM is 66 used for blank check. First it reads address DD in the EEPROM, then compares it with 66. If they are equal, the software will skip the EEPROM program procedure. If they are not equal, the software will program the EEPROM. This means that monitor makers can define their own factory modes by programming the EEPROM in advance.

The following programs are endless loops. Please refer to the main flow chart (Figure 5). There are three tasks in the endless loop.

The first task is Function Selection, basically a keyboard process program.

The second task is Mode Detection, which includes search mode and change mode.

The third task is Mode and Fucntion Display, which includes flash mode display.

(BCM) 87C751 Specification for a bus-controlled monitor

Mode Detection

Beginning with branch "B", Mode Detection Flow Chart (Figure 7), the block at the top of the flow chart is "Read Previous Mode" (the time before 0.5 second ago) and includes Horizontal Sync Frequency, Vertical Sync Frequency, Horizontal Sync Polarity, and Vertical Sync Polarity. The second block is a comparison test block. When current mode (from 0.5 second ago until now) parameters are the same as those in previous mode, the program will branch to the right test block. Since the mode is not changed, the second test block in the right part of the flow chart will branch to leave the Mode Detection section.

If the current mode (from 0.5 second ago until now) parameters are not the same as the previous mode (the time before 0.5 second ago), the first test block from the top of the flow chart will branch to search all mode parameters in the EEPROM to find out what the current mode should be. The left loop of the flow chart checks for the end of the search procedure, i.e., if all modes in the EEPROM are searched and checked, and the outcome is the same, then this test block will branch to set up a new user mode (19), as per the 4 steps indicated in the central flow chart line.

The first step in setting up a new user mode is to "Put New Parameters" (such as Horizontal Sync Frequency, Vertical Sync Frequency, Horizontal Sync Polarity, and Vertical Sync Polarity) into the EEPROM. The new mode parameters are always saved in the last mode address. If the configuration allowing 10 user modes is selected, then diode 2 is added. If one was found to be the same, the program will branch to the right test block. If it then finds that there is a mode change, it will branch to Reload Mode Data to DAC to complete the mode change procedure.

When the mode change procedure is completed, the monitor will be working in a new mode. Since the program enters the Mode Detection task every 0.5 second, it takes from 0.5 to 1.0 second to finish the change of mode. To save the new user mode (mode 19) to other user mode (10–18), the user can use the RELOAD key to save the new user mode to other user mode (modes 10 through 18).

If the configuration for 10 user modes is selected, it is highly recommended that you save new user mode to other user mode (10–18) because the last mode "mode 19", will be overwritten by any new user mode whenever a new user mode is detected, after new user mode parameters are detected.

The second step in setting up a new user mode is "Calculating VCO Output Value" (see Figure 8). There are two different curves for the designer to select. If diode 1 is removed, the VCO Output Voltage will be in direct ratio to the Horizontal Sync Frequency. If diode 1 is added, the VCO Output Voltage will have an indirect ratio.

The third step in setting up a new user mode is "Put VCO Value to EEPROM".

The last step is "Reload Mode Data to DAC". After reloading the DAC, the monitor is changed to the new user mode.

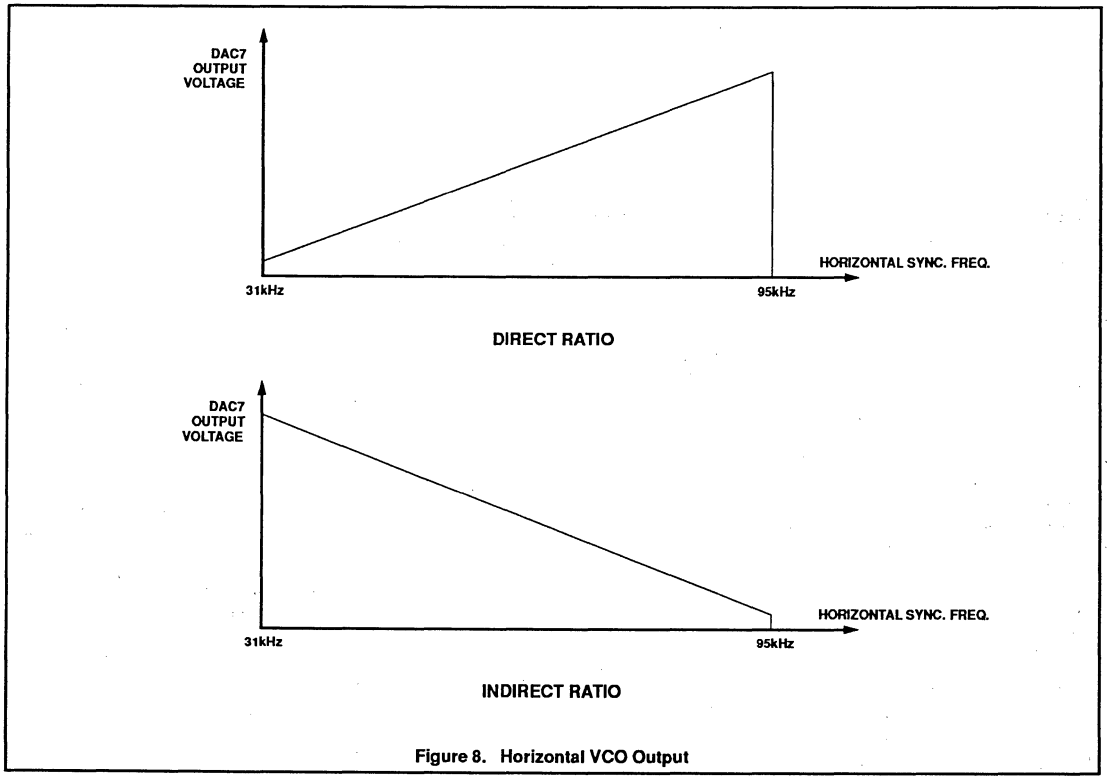


Figure 8. Horizontal VCO Output

(BCM) 87C751 Specification for a bus-controlled monitor

AN442

Key Function Selection

The first task is Function Selection (see Figure 6). When the program scans the keyboard, it first checks for the function key. If it is depressed, the program will branch right to change function VALUE(+1). the function LEDs are lit in sequence, i.e., when the second LED is lighting and the program detects a press in function key, the program will light the third LED and turn off the second LED. If the program detects that the last LED is lighting, it will turn on the first LED and turn off the last LED.

If the key pressed by the user is not a function key, the program will check if it is the reload key. If it is not, then the second test block will branch left to check Up/Down keys. both the Up and Down keys have two different definitions. When the user is updating function contents, Up and Down keys are used to change the function contents stored in the EEPROM. In that situation, the flash flag is not set; after the program branch left from the reload key test block, the program will test flash flag, then the program will change the output of DAC.

If a flash flag is set (see next paragraph, also), Up and Down keys will change the flashing mode displayed on two seven-segment LED displays. The flashing mode is valid between 10 and 19. The Up key will add one to the flashing mode unless the flashing mode is already 19, while the Down key will subtract one unless the flashing mode is already 10.

If the reload key is pressed while the program is in last mode (19), i.e., a new mode which is not the same as any mode entered in the EEPROM, the program will check to see if this is the first press. If this is the first press, the fouth test block in the middle will branch right to set the flash flag, after the flash flag is set, the mode displayed on the two seven-segment LED displays will start to flash.

Between the first press of the reload key and the second press of the reload key, the Up and Down keys can be used to change the falshing mode to a destination mode. When the program detects the second press of the reload key, the program will execute all the center blocks in the flow chart. First, starting to clear the flash flag; second, to get last mode data from EEPROM; and third, to put the mode data into a new address. The desitnation is decided by the user via Up and Down keys to select flashing mode, then pressing the reload key to make the flashing mode be a still mode. After mode display is still, the user finishes defining the destination user mode. This destination user mode will last forever unless the reload key is used to redefine it.

If the user presses the reload key, and the program is not in the last mode, it will branch right to reset mode data in EEPROM, starts to read original mode data in microcontroller's PROM, then puts these data to EEPROM, the outputs to DAC. This function is to help the user to restore the monitor when the monitor display is out of control. For example, if the user adjusts the Horizontal phase too broad, then monitor may become out of sync. As a consequence the screen will be a mess, and it is not easy for the user to re-adjust for correction. This feature will minimize possible complaints from customers.

CIRCUIT DESCRIPTION

U1–87C751 is an 8-bit microcontroller and the heart of the Bus-Controlled Monitor. The 87C751 receives Vertical Sync and Horizontal Sync signals from pins P1.5 and P1.6. The R3,C5 in pin P1.5 is a low pass protection circuit. It can prevent the Vertical Sync signals, including Horizontal Sync Pulse, from interferring with the counting of the Vertical Sync Frequency. The R2 in pin P1.6 is only used for protection of 87C751.

The 87C751 automatically checks the mode parameters from these two pins, then switches the DAC from the old mode to a new mode. When 87C751 is checking the mode, it reads different mode parameters from the EEPROM via I^2C bus, then decides whether there has been a change in mode. If a change is needed, the 87C751 will first mute video by sending a LOW to pin P0.2, then reads the correct mode data from the EEPROM via the I^2C bus. It then puts the data to the DAC via I^2C bus. Because of the I^2C bus, the connections between the EEPROM, DAC, and the microcontroller are very simple.

The system clock is provided by a 12MHz crystal connected to Pins 10, 11 of the 87C751. Basically, port1 is used for input, but P1.4 and P1.7 are used for output. P1.0 to P1.3 are used for keyboard and configuration input.

Port0 has only three pins; P0.0 and P0.1 are used for I^2C control. P0.2 is an output pin used to test configurations. Port3 is basically used for output; P3.0 to P3.4 are used for mode seven-segment LED display output; P3.5 to P3.7 are used for funciton display. Both mode display and function display need extra decoders, an HEF4511B seven-segment display driver is used to display the mode, while an HEF4556B dual 2-to-4 decoder is used to display function LEDs.

PCF8582 is a 2k-bit EEPROM. R4,C1 constructs an external R-C time to program the EEPROM. In normal operation the EEPROM needs 30ms to program one byte. C2 is a decoupling capacitor to stabilize the DC supply voltage for PCF8582.

TDA8444 is an octal 6-bit DAC. R7,VR5,C# constructs a reference voltage to define the DAC's maximum output voltage. In practice, the reference voltage must be below 10.5 volts, so R7 is added to prevent the reference voltage from exceeding that limit. C4 is also a decoupling capacitor to stabilize the DC supply voltage for TDA8444.

The upper half of the HEF4556 is used to provide a switch signal to select Horizontal OSC time constant capacitors. The four outputs are Active-LOW. When the Horizontal Sync Frequency (H.S.F.) falls in one of the four ranges, the corresponding output pin will go low. The four ranges are:

(H.S.F. < 35kHz),
(35kHz < H.S.F. < 40kHz),
(40kHz < H.S.F. < 50kHz),
(H.S.F. > 50kHz).

The enable input in the upper part of the HEF4556B can be used to extend the upper limit of the switch signal. The lower part of the HEF4556B is used to display function LEDs, the fifth LED is driven by an NPN transistor. When the transistor is turned on by the microcontroller, it also disables the lower part of the HEF4556B. If multiple Up/Down keys are configured, the function LEDs are replaced by five pairs of Up/Down keys.

Because the tenth digit of the mode display is either "1" or blank, the driver of the tenth digit uses only one transistor. The driver of the base digit employs an HEF4511B, it is a BCD seven-segment display driver with output Active-HIGH, so only common cathode types can be used. The factory can use one LED to replace a BCD display. When the LED is lit, the BCD display can display user mode, or factory mode. This is one way to reduce the system cost.

If multiple Up/Down keys are not configured, the keyboard has four keys:

Function key,
Up key
Down key
Reload key.

If multiple Up/Down keys are configured, there will be no function key, but five pairs of Up/Down keys and one Reload key. the multiple Up/Down keys are configured by adding a diode in JP7. If adding a diode in JP6, there will be only four functions available, i.e., there will be only four paris of Up/Down keys or four function LEDs.

LM7805 is a power regulator IC, it changes 12V to 5V to supply the whole circuit except TDA8444, which uses a 12V power supply to provide a wider range of DAC output.

(BCM) 87C751 Specification for a bus-controlled monitor

There is a table to explain the usage of each pin in the JP1 socket. JP2 and JP8 are connected together, JP3 is only used for future automatic alignment (including production line) if necessary. JP4 to JP7 are used to select hardware configurations as previously mentioned.

SPECIFICATION OF THE SYSTEM

1. The input signals to the system are Horizontal Sync and Vertical Sync. The system accepts standard TTL level signals, i.e., $V_{IH} > 2V$, and $V_{IL} < 0.4V$. Horizontal Sync tolerance is ±0.5kHz, and Vertical Sync tolerance is +2/–2 Hz.
2. There are eight DAC output signals. Their maximum output voltage can be preset by setting a voltage on the TDA8444's V_{MAX} pin. The voltage on the V_{MAX} pin must be below 10.5V and also below the voltage on the TDA8444's V_P pin. For other detailed output current characteristics, please refer to Philips data books IC02a, IC02b and *80C51 and Derivative Microcontrollers.*
3. The Horizontal switch outputs have four pins. they are standard CMOS B-type buffered outputs. They are Active-LOW, i.e., there will be only one output active at any time. If the designers wants to add ranges in higher Horizontal Sync Frequency, the designer can put extra circuits onto the demo board. For example, an OPA can be added as a comparator to detect the VCO output. If the VCO output is higher than a certain voltage (VCO's 60kHz output voltage), the OPA will be triggered and the upper half of the HEF 4556 can be disabled by the OPA via HEF4556's Pin 1, when HEF4556 is disabled, the four Horizontal switch outputs will remain HIGH, then the OPA's output can be used as another switch output.
4. Total current consumption is around 25–90mA, depending on the number of LED and seven-segment displays being lit.

PARTS LIST

87C751 BUS CONTROLLED MONITOR
TH-9102/4
Bill of Materials November 7, 1991

Revised: November 7, 1991
Revision:
12:08:46 Page 1

ITEM	QUANTITY	REFERENCE	PART
1	1	C1	2700pF
2	6	C2, C3, C4, C8, C12, C13	0.1µF
3	1	C5	0.01µF
4	1	C6	100µF
5	2	C7, C11	1µF
6	2	C9, C10	33pF
7	5	D1, D2, D3, D4, D5	LED
8	3	JP1, JP2, JP8	Header 16
9	1	JP3	Header 4
10	4	JP4, JP5, JP6, JP7	Jumper (add diode)
11	2	Q1, Q2	BC548
12	1	R1	470R*7
13	11	R2, R3, R8, R9, R10, R11, R14, R15, R17, R19, R20	470R
14	6	R4, R6, R12, R13, R16, R18	22K
15	1	R5	VR10K
16	1	R7	2K
17	2	R21, R22	56R
18	5	S1, S2, S3, S4, S5	SW Pushbutton
19	1	U1	87C751
20	1	U2	HEF4556B
21	1	U3	PCF8582
22	1	U4	TDA8444
23	1	U5	HEF4511B
24	1	U6	LM7805
25	2	U7, U8	DISP-7
26	1	Y1	12MHz

(BCM) 87C751
Specification for a bus-controlled monitor

AN442

PARTS LIST

87C751 BUS CONTROLLED MONITOR
TH-9102/5
Bill of Materials November 13, 1991

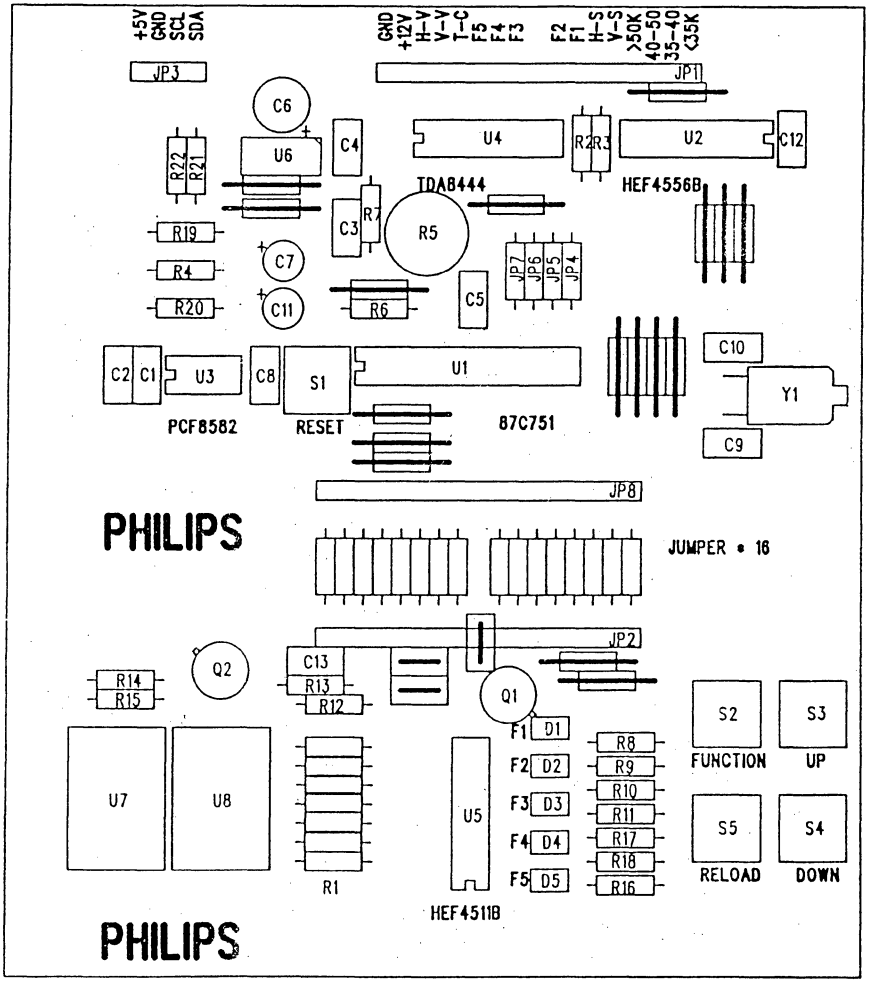
Revised: November 13, 1991
Revision:
15:42:31 Page 1

ITEM	QUANTITY	REFERENCE	PART
1	1	C1	2700pF
2	6	C2, C3, C4, C8, C12, C13	0.1µF
3	1	C5	0.01µF
4	1	C6	100µF
5	2	C7, C11	1µF
6	2	C9, C10	33pF
7	3	JP1, JP2, JP8	Header 16
8	1	JP3	Header 4
9	4	JP4, JP5, JP6, JP7	Jumper (add diode)
10	2	Q1, Q2	BC548
11	1	R1	470R*7
12	10	R2, R3, R8, R9, R10, R11, R14, R15, R19, R20	470R
13	6	R4, R6, R12, R13, R16, R18	22K
14	1	R5	VR10K
15	1	R7	2K
16	2	R21, R22	56R
17	12	S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12	SW Pushbutton
18	1	U1	87C751
19	1	U2	HEF4556B
20	1	U3	PCF8582
21	1	U4	TDA8444
22	1	U5	HEF4511B
23	1	U6	LM7805
24	2	U7, U8	DISP-7
25	1	Y1	12MHz

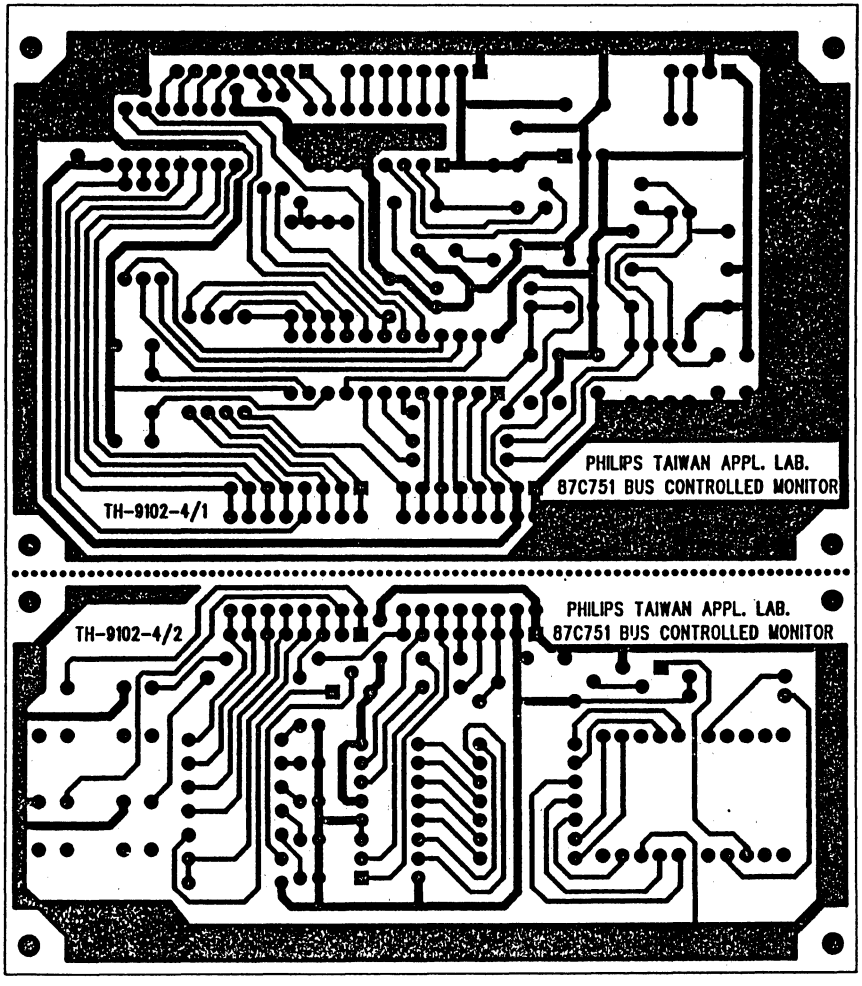
(BCM) 87C751
Specification for a bus-controlled monitor

AN442

CIRCUIT DIAGRAM



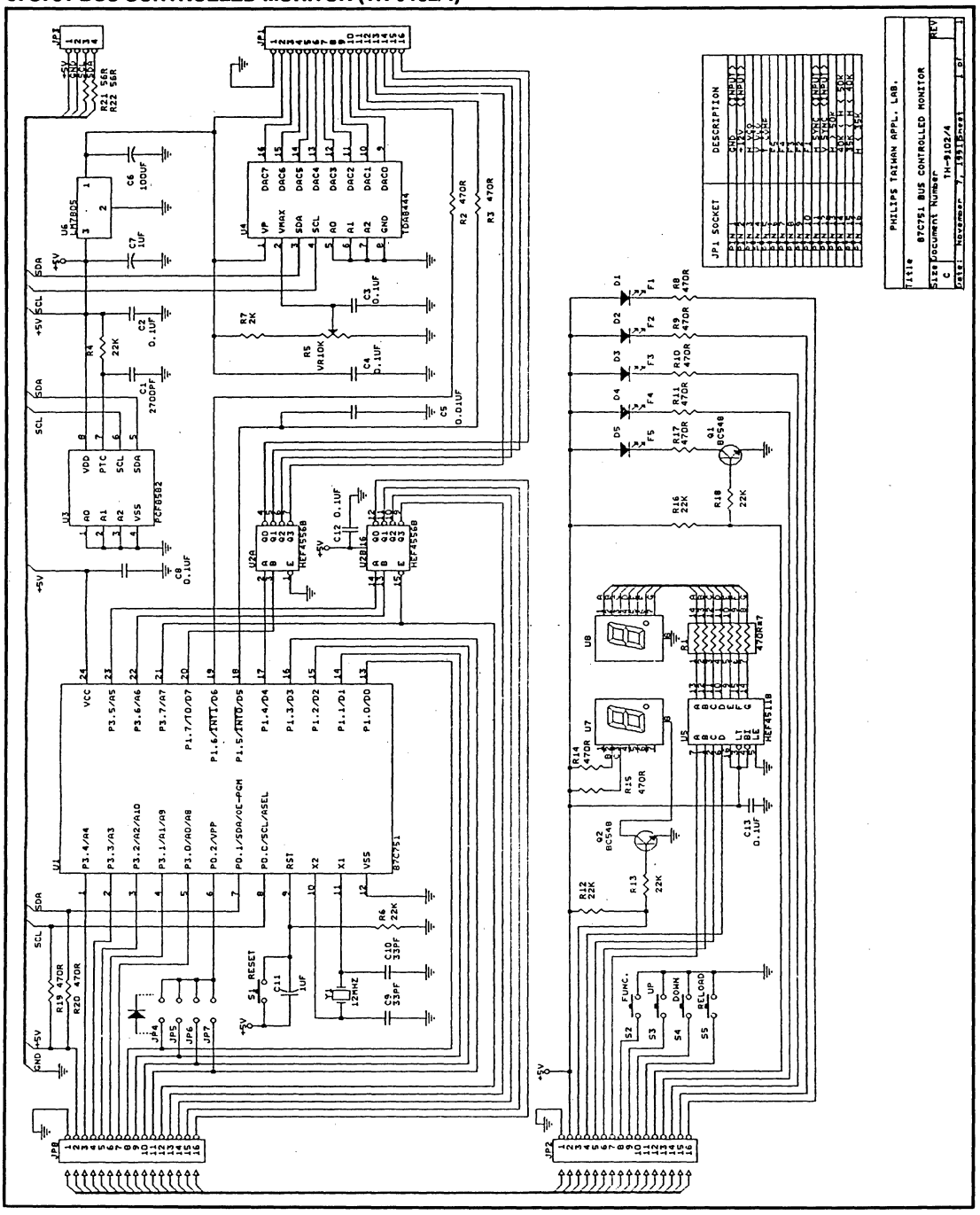
PRINTED CIRCUIT BOARD



(BCM) 87C751
Specification for a bus-controlled monitor

AN442

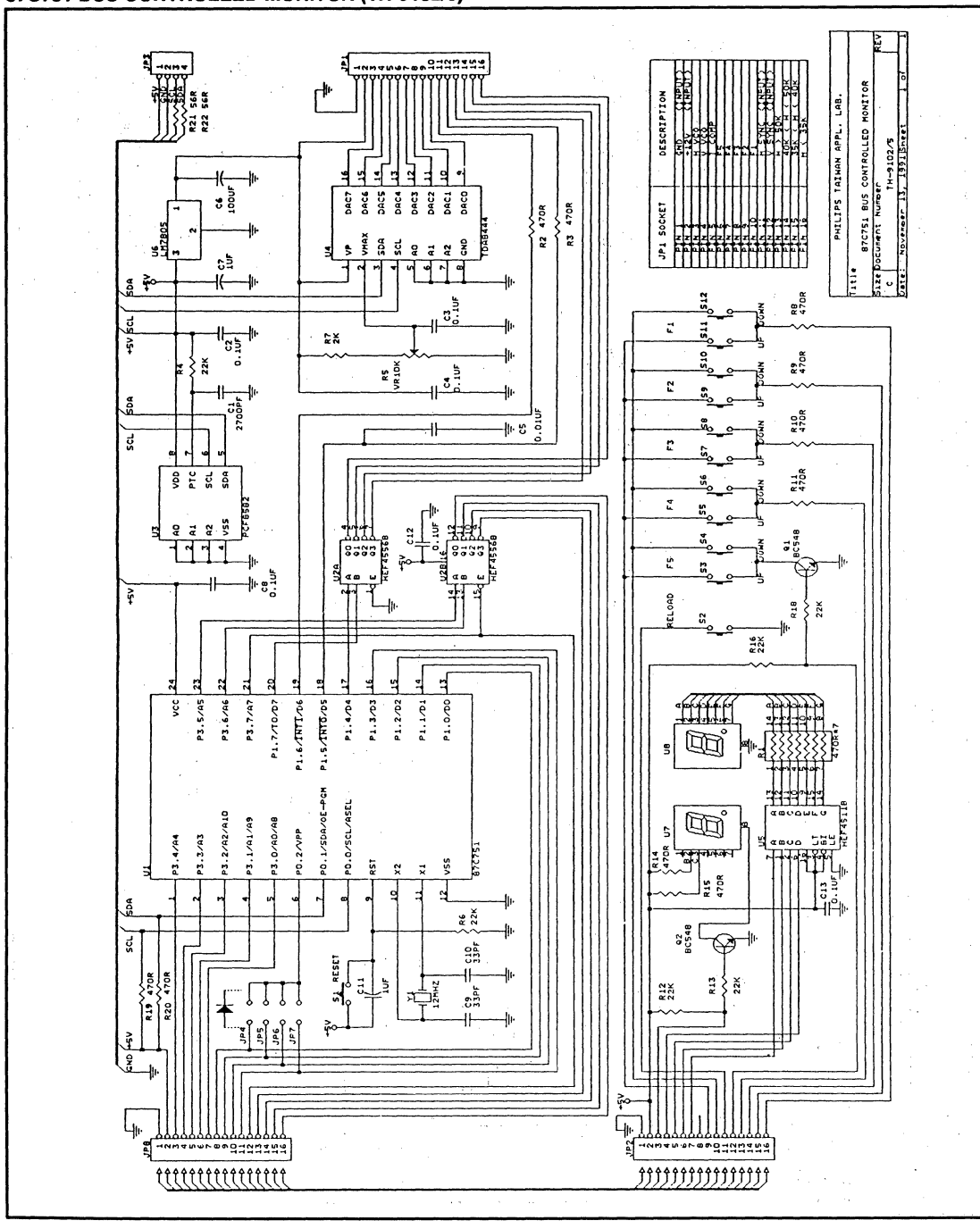
87C751 BUS CONTROLLED MONITOR (TH-9102/4)



(BCM) 87C751
Specification for a bus-controlled monitor

AN442

87C751 BUS CONTROLLED MONITOR (TH-9102/5)



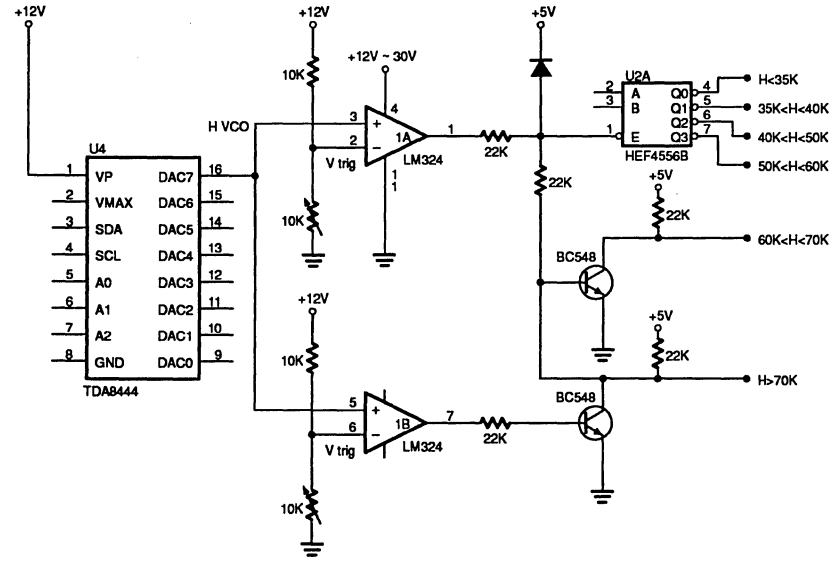
(BCM) 87C751 Specification for a bus-controlled monitor

AN442

APPENDIX A
References

1. Philips Data Book IC02a, IC02b
   Video and Associated Systems
2. Philips Data Book IC20
   80C51 and Derviative Microcontrollers
   Title: "AN422: Using the 8XC751 Microcontroller as an I²C bus Master"
3. Philips Data Book
   RF Communications
   Title: "AN168: The Inter-Integrated (I²C) Serial Bus: Theory and Practical Consideration",
   Author: Carl Fenger
4. Title: "ETV8831: Deflection Processor TDA8433 with I²C Control"
   Author: DJA Teuling
5. Title: "ETV89008: VGA Monitor with the High Resolution Colour Tube M34ECL10X36"
   Author: H. Nerhees

APPENDIX B



Example Circuit to Expand More Ranges in Cs Selection

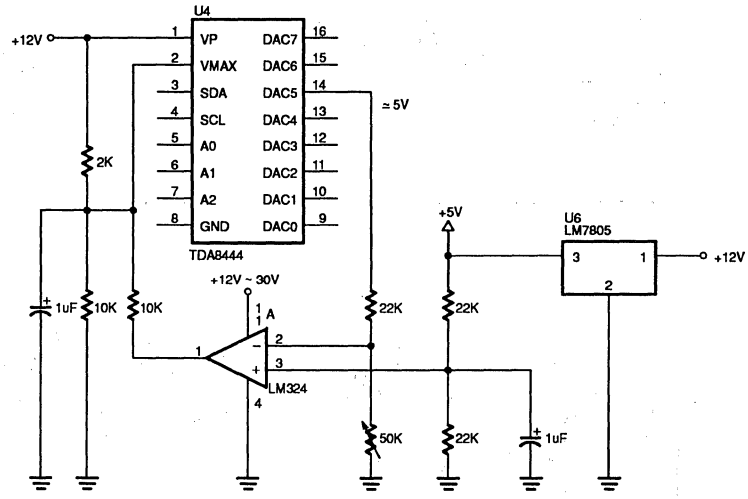
When you want to determine Vtrig signal, please disconnect Pulse Signal Generator (P.S.G.) Output to H. Sync demoboard from monitor and connect, and use input, set P.S.G. to 60kHz, TTL level output, then power on the demoboard, the mode display should display "19", the the voltage in DAC H-V output pin is the trigger level voltage of 60kHz.

You can set P.S.G to 70kHz to measure trigger level voltage of 70kHz.

(BCM) 87C751 Specification for a bus-controlled monitor

AN442

APPENDIX C



Example Circuit of Temperature Compensation in TDA8444

All VRs in this application note can be charged to fix resistors, when proper dividing voltage is determined.

Section 6
Development Support Tools

CONTENTS

Development Support Tools 869

Nohau EMUL51-PC — PC-Based In-Circuit Emulator 876

Metalink 886

DB-51 CEIBO Devleopment Board 890

SDS 8051 Stand-alone Debug Station 892

OM4142 (ASM51) Cross-Assembler Package 898

OM4144 (PLMTI51) Compiler Package 900

OM4136 8051 C Cross-Compiler 903

OM4129 Symbolic Debugging Package XRAY51 905

Development support tools

SECTION 6

DEVELOPMENT SUPPORT TOOLS

Philips Semiconductors manufactures support tools and also works closely with many "third-party" vendors who provide support tools for our wide variety of 80C51-based microcontroller derivatives.

Development Systems

In most cases, development systems are available in two versions for ROM and ROMless applications. The ROM emulation products are capable of supporting all versions of a given device type, including EPROM, ROM, and ROMless devices. In contrast, a ROMless emulator can only support applications designed for a ROMless microcontroller. Most development systems are designed to connect to an IBM-PC or compatible personal computer.

EPROM Programming Support

Philips Semiconductors works closely with major suppliers of EPROM programming equipment to support our family of EPROM microcontrollers. As a result, EPROM programming support is available within the programming facilities of many major distributors.

The following is a list of vendors that offer support for Philips Semiconductors 80C51 microcontroller family.

DEVELOPMENT SYSTEM CONTACTS

COMPANY	ADDRESS	TELEPHONE
Ashling Microsystems Limited	Plassey Technological Park Limerick, Ireland	(353) 63-334466
BSO Tasking	128 Technology Center P.O. Box 9164 Waltham, MA 02254-9164	(617) 894-7800
Ceibo Ltd.	105 Gleason Rd. Lexington, MA 02173 Merkazim Building, Industrial Zone P.O. Box 2106 Herzelia 46120, ISRAEL	(617) 863-9927 972-52-555387
Nohau Corp.	51 E. Campbell Ave. Campbell, CA 95008	(408) 866–1820
MetaLink Corp.	325 E. Elliot Road, Suite 23 Chandler, AZ 85225	(602) 926–0797
Philips Semiconductors	Corporate Centre Building BAE-2 P.O. Box 218 5600 MD Eindhoven The Netherlands	31-40-724223
SIGNUM Systems	171 E. Thousand Oaks Blvd., #202 Thousand Oaks, CA 91360	(805) 371-4608

EPROM PROGRAMMING SUPPORT CONTACTS

COMPANY	ADDRESS	TELEPHONE
Advin Systems	1050-L East Duane Ave. Sunnyvale, CA 94086	(408) 736-2503
BP Microsystems	10681 Haddington #190 Houston, TX 77043	(800) 225-2102 (713) 461-9430
Data I/O Corp.	10525 Willows Road N.E. P.O. Box 97046 Redmond, WA 98073-9746	(206) 867-6899
Logical Devices, Inc.	1201 Northwest 65th Place Ft. Lauderdale, FL 33309	(305) 974-0967
Logical Systems	P. O. Box 6184 Syracuse, NY 13217-6184	(315) 478-0722
Needham's Electronics	4535 Orange Grove Ave. Sacramento, CA 95841	(916) 924-8037
North Valley Products	P.O. Box 32899 San Jose, CA 95152	(408) 929-5345
Strebor Data Communications	1008 N. Nob Hill American Fork, UT 84003	(801) 756-3605

SOFTWARE SUPPORT CONTACTS

COMPANY	ADDRESS	TELEPHONE
Franklin Software, Inc.	888 Saratoga Ave. #2 San Jose, CA 95129	(408) 296–8051
Archimedes Software, Inc.	2159 Union St. San Francisco, CA 94123	(415) 567–4010
BSO/Tasking	Tasking Software BV P.O. Box 899 3800 AW Amersfoort The Netherlands BSO Tasking 128 Technology Center P.O. Box 9164 Waltham, MA 02254-9164	31-33-55-85-84 (Telephone) 31-33-55-00-33 (Fax) (617) 894-7800 (Telephone) (617) 894-0551 (Fax) (710) 324-0760 (Telex) (800) 458-8276 (Toll Free)

MICROCONTROLLER DEVELOPMENT SYSTEMS

PRODUCT	DEVICES SUPPORTED
NOHAU CORPORATION	
EMUL51–PC/E32	12MHz Emulator, 32k emulation memory
EMUL51–PC/E128	12MHz Emulator, 128k emulation memory
EMUL51–PC/E32–16	16MHz Emulator, 32k emulation memory
EMUL51–PC/E128–16	16MHz Emulator, 128k emulation memory
EMUL51–PC/128–20	20MHz Emulator, 128k emulation memory
EMUL51–PC/128–24	24MHz Emulator, 128k emulation memory
EMUL51–PC/128–30	30MHz Emulator, 128k emulation memory
EMUL51–PC/128–BSW	12MHz Emulator, 128k bankswitched CODE memory
POD–054	12MHz 83C053, 83C054, 87C054 pod
POD–31	12MHz 8031 pod
POD–C31	12MHz 80C31 pod
POD–C31–1	16MHz 80C31 pod
POD–C31–20	20MHz 80C31 pod
POD–C31–24	24MHz 80C31 pod
POD–C31–30	30MHz 80C31 pod
POD–32	12MHz 8032 pod
POD–C32	12MHz 80C32 pod
POD–C32–16	16MHz 80C32 pod
POD–C652	12MHz 80C652 pod
POD–C652–16	16MHz 80C652 pod
POD–C51B	12MHz bondout pod for 8051, 80C51, 83C552, 83C652, 83C654, 83C851, and EPROM or ROMless versions of the above
POD–C51B–16	16MHz bondout pod for 8051, 80C51, 83C552, 83C652, 83C654, 83C851, and EPROM or ROMless versions of the above
POD–C51B–24	24MHz version of the above
POD–C52	12MHz 80C32, 80C52, 87C52
POD–C52–16	16MHz 80C32, 80C52, 87C52
POD–C451–DIP	12MHz 80C451 DIP pod
POD–C451–DIP–16	16MHz 80C451 DIP pod
POD–C451–PGA	12MHz 80C451 PLCC pod (PGA from pod)
POD–C451–PGA–16	16MHz 80C451 PLCC pod (PGA from pod)
POD–C451B–PGA	12MHz bondout pod for 83C451, 87C451, 80C451 PLCC (PGA from pod)
POD–C528	12MHz 83C528, 87C528
POD–C528–16	16MHz 83C528, 87C528
POD–C550–PGA	12MHz 80C550, 83C550, 87C550
POD–C552–PGA	12MHz 80C552 PLCC (PGA from pod)
POD–C552B–PGA	12MHz bondout pod for 83C552, 87C552, 80C552 PLCC (PGA from pod), 80C562, 83C562
POD–C552B–PGA–16	16MHz bondout pod for 83C552, 87C552, 80C552 PLCC (PGA from pod), 80C562, 83C562
POD–C592–PGA	12MHz 80C592, 83C592, 87C592 pod

MICROCONTROLLER DEVELOPMENT SYSTEMS (Continued)

PRODUCT	DEVICES SUPPORTED
NOHAU CORPORATION (Continued)	
POD–C652B	Order as POD–C51B
POD–C851B	Order as POD–C51B
POD–C751	12MHz 83C751, 87C751 pod
POD–C751–16	16MHz 83C751, 87C751 pod
POD–C752	12MHz 83C752, 87C752 pod
POD–C752–16	16MHz 83C752, 87C752 pod
EMUL51–PC/TR4	12MHz 4k trace buffer option
EMUL51–PC/TR16	12MHz 16k trace buffer option
EMUL51–PC/TR4–16	16MHz 4k trace buffer option
EMUL51–PC/TR16–16	16MHz 16k trace buffer option
EMUL51–PC/TR16–20	20MHz 16k trace buffer option
EMUL51–PC/TR16–24	24MHz 16k trace buffer option
EMUL51–PC/TR16–30	30MHz 16k trace buffer option
EMUL51–PC/BOX–S	Box with serial port and cable. Box allows operation of emulator external to PC.
EMUL51–PC/BOX–M	Box with serial port and modem
EMUL51–PC/BOX–CS	Serial box with emulator (E128–16) and trace (TR16–16)
EMUL51–PC/BOX–CS–20	Serial box with emulator (E128–20) and trace (TR16–20)
EMUL51–PC/BOX–CS–24	Serial box with emulator (E128–24) and trace (TR16–24)
EMUL51–PC/BOX–CS–30	Serial box with emulator (E128–30) and trace (TR16–30)
METALINK CORPORATION	
IM–8051/200–20	iceMASTER–8051 emulator Model 200, 32K emulation memory, 20MHz
IM–8051/400–20	iceMASTER–8051 emulator Model 400, 32K emulation memory, 4K trace buffer, 2 performance analyzers, 20MHz
IM–8051/400–24	iceMASTER–8051 emulator Model 400, 128K emulation memory, 4K trace buffer, 2 performance analyzers, 24MHz
128KUP	128K memory expansion option for iceMASTER–8051
752/1 PGMPC	Programmer accessory for iceMASTER to program 87C751, 87C752
8031–12PC	0.5 to 12MHz 8031, 80C31
8031–16PC	0.5 to 16MHz 8031, 80C31
8031–20PC	0.5 to 20MHz 8031, 80C31
8031–24PC	0.5 to 24MHz 8031, 80C31
8032–12PC	0.5 to 12MHz 8031, 80C31, 8032, 80C32
8032–16PC	0.5 to 16MHz 8031, 80C31, 8032, 80C32
8032–20PC	0.5 to 20MHz 8031, 80C31, 8032, 80C32
8032–24PC	0.5 to 24MHz 8032, 80C32, 8031, 80C31
8052–12PC	0.5 to 12MHz 8031, 80C31, 8032, 80C32, 8051, 8751, 80C51, 87C51, 8052, 8752, 80C52, 87C52
8052–16PC	0.5 to 16MHz 8031, 80C31, 8032, 80C32, 8051, 8751, 80C51, 87C51, 8052, 8752, 80C52, 87C52
80410–12PC	0.5 to 12MHz 80CL410
80451–12PC	1.2 to 12MHz 80C451
80451–16PC	1.2 to 16MHz 80C451
80528–12PC	1.2 to 12MHz 80C528
80528–16PC	1.2 to 16MHz 80C528
80552–12PC	1.2 to 12MHz 80C552, 80C562
80552–16PC	1.2 to 16MHz 80C552, 80C562
80652–12PC	1.2 to 12MHz 8031, 80C31, 80C652
80652–16PC	1.2 to 16MHz 8031, 80C31, 80C652
80851–12PC	1.2 to 12MHz 8031, 80C31, 80C851
83053–12PC	6 to 12MHz 83C053, 83C054, 87C054
83451–12PC	1.2 to 12MHz 80C451, 83C451, 87C451
83528–12PC	1.2 to 12MHz 80C528, 83C528, 87C528, 83C524, 87C524
83528–16PC	1.2 to 16MHz 80C528, 83C528, 87C528, 83C524, 87C524
83550–10PC	1.2 to 10MHz 80C550, 83C550, 87C550
83552–12PC	1.2 to 12MHz 80C552, 83C552, 87C552, 80C562, 83C562
83552–16PC	1.2 to 16MHz 80C552, 83C552, 87C552, 80C562, 83C562
83652–12PC	1.2 to 12MHz 80C652, 83C652, 87C652, 80C552, 83C552, 87C552, 80C562, 83C562
83652–16PC	1.2 to 16MHz 80C652, 83C652, 87C652, 80C552, 83C552, 87C552, 80C562, 83C562

MICROCONTROLLER DEVELOPMENT SYSTEMS (Continued)

PRODUCT	DEVICES SUPPORTED
METALINK CORPORATION (Continued)	
83654–12PC	1.2 to 12MHz 80C652, 83C652, 87C652, 83C654, 87C654, 80C552, 83C552, 87C552, 80C562, 83C562
83654–16PC	1.2 to 16MHz 80C652, 83C652, 87C652, 83654, 87C654, 80C552, 83C552, 87C552, 80C562, 83C562
83751–12PC	0.5 to 12MHz 83C751, 87C751
83751–16PC	0.5 to 16MHz 83C751, 87C751
83752–12PC	0.5 to 12MHz 83C752, 87C752
PHILIPS SEMICONDUCTORS	
OM4120S	SDS stand-alone debug station (20MHz)
OM1079	8XCL410/710, 8XCL51 DIP emulator probe
OM1092 & OM1095	8XC552/562 PLCC emulator probe
OM1092 & OM1096	8XC652/654/51 DIP/PLCC emulator probe
OM1092 & OM1097	8051/80C51 DIP/PLCC emulator probe
OM1092	8XC851 DIP/PLCC emulator probe
OM1094	8XC751 DIP/PLCC emulator probe
OM4123	8XC451 PLCC emulator probe
OM5054	8XC053/054 SDIP emulator probe
OM5072	8XC752 DIP emulator probe
OM4124	Probe adapter: 68-pin PLCC probe to 64-pin DIP socket for 8XC451
OM4125	Probe adapter: 40-pin DIP probe to 44-pin PLCC socket for 80C51, 80CL51
OM1095	8XC552/562 converter for OM1092
OM1096	8XC652/654 converter for OM1092
OM1097	80C31/80C51 converter for OM1092
OM4111 & OM4110	8XC528 (also 80C32/52) DIP/PLCC emulator probe
OM4112 & OM4110	8XC592 PLCC emulator probe
OM4118/1	Contactless card reader adaptor for OM4119
OM4118/2	ISO contact card reader adaptor for OM4119
OM4118/3	AFNOR contact card reader adaptor for OM4119
OM4119	83C852 emulator probe
OM4129	XRAY51 symbolic debug package for SDS
OM4142	80C51 family Cross Assembler for MS-DOS
OM4144	PLM-51 compiler for MS-DOS
OM4136	C language 8051 compiler for MS-DOS

NOTES:
1. These items are not available directly from Signetics, but may be purchased in the U.S. from BSO/Tasking (see contact list).
2. All emulator probes are 16MHz except OM4123, which is 12MHz.

EPROM MICROCOMPUTER PROGRAMMING SUPPORT

DEVICE	MANUFACTURER/MODEL	MODULE/ADAPTOR	SOFTWARE VERSION
87C054 SDIP	N. Valley Products Signetics/Ceibo MP-51	SAM-054SD PPA-054SD	
87C51 DIP	Advin Sailor-PAL/SA, /SB BP Microsystems EP-1140 Ceibo MP-51 (PMP51SD) Data I/O Unisite 40 Data I/O Unipak 2b Data I/O Series 1000 Logical Devices ALLPRO N. Valley Products SPGM–100 Philips LCPX5X40 (P8051LCP40) Strebor PLP-S1A	Adaptor-8751 PPA-51XSD 351B103 SR40 SAM–51SD MC4851DIP	 V2.2 V16 V05 (Use Intel 87C51 menu) V1.47 V1.0
87C51 PLCC	Ceibo MP-51 (PMP51SD) Data I/O Unisite 40 Data I/O Unipak 2b Logical Devices ALLPRO N. Valley Products SPGM–100 Philips LCPX5X40 (P8051LCP40)	PPA-51XSD Chipsite 351B103P Required SAM–51ASD	 V2.3 V16 V1.47 V1.0
87C52 DIP	BP Microsystems EP-1140 Ceibo MP-51 (PMP51SD) Data I/O 29B, Unipak 2b Data I/O Unisite 40 N. Valley Products SPGM–100 Philips LCPX5X40 (P8051LCP40)	 PPA-XSD 351B103 SAM–52SD	 V23 V3.1
87C451 DIP	Ceibo MP-51 (PMP51SD) Logical Devices ALLPRO N. Valley Products SPGM–100	PPA-451SD PPRequired SAM–451SD	 V1.47 V1.0
87C451 PLCC	Advin Sailor-PAL/SA, /SB Ceibo MP-51 (PMP51SD) N. Valley Products SPGM–100 Data I/O Unisite Philips LCPX5X (P8051LCPX)	Adaptor-87451 PPA-451ASD SAM–451ASD Chipsite	 V1.0 V2.8
87C528 DIP	BP Microsystems EP-1140 Ceibo MP-51 N. Valley Products SPGM–100 Philips LCPX5X40 (P8051LCP40)	 PPA-XSD SAM–528	
87C528 PLCC	Ceibo MP-51 (PMP51SD) N. Valley Products SPGM–100 Philips LCPX5X40 (P8051LCP40)	PPA-51XSD SAM–528A	
87C552	Ceibo MP-51 (PMP51SD) N. Valley Products SPGM–100 Data I/O Unisite Philips LCPX5X (P8051LCPX)	PPA-552ASD SAM–552ASD Chipsite	 V2.2 V3.1
87C652 DIP	BP Microsystems EP-1140 Ceibo MP-51 N. Valley Products Philips LCPX5X40 (P8051LCP40)	 PPA-51XSD SAM-52SD	
87C652 PLCC	Ceibo MP-51 (PMP51SD) Philips LCPX5X40 (P8051LCP40)	PPA-51XSD	
87C654 DIP	BP Microsystems EP-1140 Ceibo MP-51 (PMP51SD) N. Valley Products SPGM–100 Philips LCPX5X40 (P8051LCP40)	 PPA-51XSD SAM–654SD	
87C654 PLCC	Ceibo MP-51 (PMP51SD) Philips LCPX5X40 (P8051LCP40)	PPA-51XSD	

EPROM MICROCOMPUTER PROGRAMMING SUPPORT (Continued)

DEVICE	MANUFACTURER/MODEL	MODULE/ADAPTOR	SOFTWARE VERSION
87C751 DIP	Advin Sailor-PAL/SA, /SB	EM-751	
	BP Microsystems EP-1140	HEAD-40A	
	Ceibo MP-51 (PMP51SD)	PPA-751SD	
	Data I/O Unisite 40		V2.3
	Data I/O 29B, Unipak 2b	351B113D	29B V6, Unipak 2B V20
	Logical Devices ALLPRO	OPTAPC–751	V1.47
	N. Valley Products SPGM–100	SAM–751SD	V1.0
	Needham's Electronics		
	MetaLink	752/1 PGMPC	V2.6a (use with MicroICE+)
	Logical Systems (Sunshine EW–901)	PA751	
	Philips LCPX5X (P8051LCPX)		
	Strebor PLP-S1A	MC7512DIP	
87C751 PLCC	Ceibo MP-51 (PMP51SD)	PPA-51XSD	
	Data I/O Unisite 40	Chipsite	V2.6
	Logical Devices ALLPRO	Required	V1.47
	N. Valley Products SPGM–100	SAM–751ASD	V1.0
87C752 DIP	Advin Sailor-PAL/SA, /SB	EM-751	
	BP Microsystems EP-1140	HEAD-40A	
	Ceibo MP-51 (PMP51SD)	PPA-752SD	
	Data I/O Unisite 40		V2.6
	N. Valley Products SPGM–100	SAM–752SD	V1.0 (Use Type 751)
	Needham's Electronics		
	MetaLink	752/1 PGMPC	V2.6a (use with MicroICE+)
	Logical Systems (Sunshine EW–901)	PA751	
	Logical Devices ALLPRO	OPTAPC-752	
	Philips LCPX5X (P8051LCPX)		
	Strebor PLP-S1A	MC7512DIP	
87C752 PLCC	Ceibo MP-51 (PMP51SD)	PPA-752ASD	
	Data I/O Unisite 40	Chipsite	V2.8
	N. Valley Products SPGM–100	SAM–752ASD	V1.0 (Use Type 751)

NOTE:
Philips programmers are available in the U.S. through Signetics distributors.

Development support tools

ADDITIONAL PROGRAMMING SUPPORT

DEVICE	MANUFACTURER	MODULE/ADAPTOR	COMMENTS
87C51/52/652/ 654/528 PLCC	Logical Systems	PA51–44	Use with any 40-pin microcontroller programming site that supports the appropriate EPROM size.
87C51/52/652/ 654/528 QFP		PA52-QFP	Use with any 40-pin microcontroller programming site that supports the appropriate EPROM size.
87C451 DIP		PA451–64	Use with any 87C51 40-pin programming site.
87C451 PLCC		PA451–68	Use with any 87C51 40-pin programming site.
87C550 DIP		550BASE	Use with any 87C51 40-pin programming site.
87C550 PLCC		PA550–44	Use with any 87C51 40-pin programming site.
87C552 PLCC		PA552–68	Use with any 8752/C52/C252/C51FA 40-pin programming site.
87C752 PLCC		PA28-28	Use with any 87C752 28-pin DIP programmer.
87C751 PLCC	Signetics	No part number assigned	This adapter allows programming the 87C751 PLCC part in conjunction with any programmer that can already program the DIP version of the part.

MICROCONTROLLER SUPPORT

PRODUCT	DEVICES SUPPORTED	MANUFACTURER	DESCRIPTION
8051 C Compiler 8051 C Compiler 80C51 C Compiler	8051 and derivatives 8051 and derivatives 8051 and derivatives	Franklin Software Archimedes Software BSO/Tasking	C Compiler for 8051 family C Compiler for 8051 family C Compiler for 8051 family
P8051DB	8051 and derivatives	Ceibo	80C51 Family Development Board
S87C00KSD	--	Signetics	I^2C Demonstration Board. 87C751 controls various I^2C peripherals. Board has sockets for 87C752, 87C652, and 87C552 also.
--	--	Signetics	The Signetics computer Bulletin Board system has available a microcontroller newsletter, application and demonstration programs for download, and the ability to send messages to microcontroler applications engineers. Access by modem at 2400, 1200, or 300 baud. The telephone numbers are: (800) 451–6644 (in the U.S.) or (408) 991–2406.
SMI–CNV451SD	80/83/87C451	Signetics	Signetics product adapts a PLCC emulator plug for the 80C451 to the DIP pinout.

nohau
CORPORATION
(408)866-1820

NOHAU EMUL51-PC – PC-based in-circuit emulator

1.0 System Architecture

Features of the Nohau EMUL51-PC in-circuit emulator include:

- Low-cost full real-time emulation
- IBM PC-bus plug-in boards or stand alone Box version with 115K baud RS232-C connection to IBM PC
- Easy-to-learn user interface with windows and pull-down menus
- Source-level debugging in C, PL/M or Pascal with full support for typed symbols
- 256k frames by 64 bit real-time trace option with time stamp
- Program Performance Analyzer

The Nohau EMUL51-PC consists of a board which plugs directly into the IBM PC/XT/AT bus for fast file transfer. An optional external box with a serial link is also available. The optional Trace board features an advanced trace function with many trigger capabilities.

The POD, which plugs into the target system, is connected with a 5 ft (1.5 m) ribbon cable to the Emulator board to provide a flexible operating range.

The EMUL51-PC uses no wait states and does not intrude on memory, stack, I/O or interrupt pins.

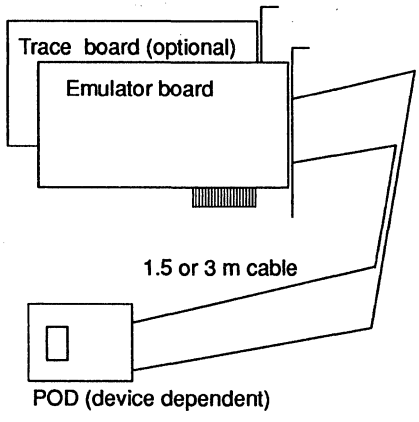


fig 1 EMUL51-PC plug-in board version

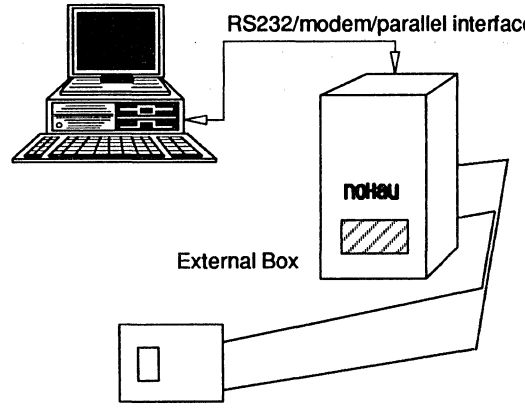


fig 2 External Box version

2.0 User Interface

The user interface is based on MS-DOS software. It incorporates a variety of techniques. The user can select to operate either with short typed commands or using pull-down menus. On-screen features include windows to monitor or alter:

- Assembly or high level language source code
- CPU registers
- Internal data and Special Function Registers
- External data
- Watch variables in C, PL/M or assembly
- Trace setup and display

All commands are supported with context-sensitive help and full on-line manual.

3.0 Specific Features

3.1 Source Level Debugging

Using high level language for code generation is a way to cut development time. Therefore the EMUL51-PC gives full support for debugging directly in C, PL/M or Pascal source code. This eliminates the need for paper listings. Breakpoints can be marked directly in the source code window. The user can single step through the program line by line and follow the program execution on the screen listing.

The trace permits the user to trace his source code in real-time.

All variables can be displayed and altered. This includes full support for typed symbols which permits the user to address such variables as floating-point, arrays and structures — both global and local.

3.2 Emulation Memory

The EMUL51-PC features 64 K of code memory and 64 K of external data memory. The addressable memory can be mapped either to target or to the emulator in 4K pages.

The emulator can load all common file formats and the user can view and alter all registers and memory areas of the microcontroller.

3.3 Breakpoints

The EMUL51-PC permits the user to generate program breakpoints in a number of ways:

- 64K program breakpoints
- 64K data read and 64K data write breakpoints
- Break on external signal
- Break on direct access to internal bit or byte memory
- Break on contents of internal register or memory
- Break on program access out-of-boundary

Using the Trace board it is possible to break on combinations of address, data, control signals, port signals and external signals.

3.4 Macros and debug session logging

Test session automation is made possible using the Macro commands. This permits the user to define his own command sequences using structures like IF/ELSE and REPEAT/WHILE. Such command sequences can be stored to a file which can be loaded automatically when the emulator is invoked.

A complete debug session and all setups can also be recorded to a file.

3.5 Trace Memory

The optional trace board features a trace buffer capable of storing up to 256k frames of 64 bit data each. The 64 bits consist of address, data, control signals, port signals, external signals and a time stamp.

The trace memory can be displayed, reprogrammed and restarted during emulation.

3.5.1 Trace Filter

The trace filter makes it possible to select what events are stored in the trace buffer. The qualifiers permit the user to define the criteria for which bus cycles are stored. The qualifiers can specify address, data, control signals, port signals and external signals.

3.5.2 Trace Trigger

The trace works much like a logic analyzer. It is therefore possible to trigger the trace on an event and display what happened before or after that event.

The trigger event can be defined using any of the qualifiers in up to eight levels. It is possible to trigger on boolean combinations of the qualifiers, or sequential combinations including a loop counter.

The trigger point can be selected anywhere within the 256k trace buffer to give full choice of pre/post trigger alignment.

3.5.3 Trace Display

The trace information can be displayed in high-level language statements, in disassembled form or in binary/hex form. It can also be stored to a file.

3.5.4 Program Performance Analyzer

With the PPA, information can be generated showing which addresses the user program spends its time on. The information can be displayed as statistics or in histogram form.

4.0 General Specifications

Host:	IBM PC/XT/AT/386/486, PS/2 or compatible with 640K of RAM. Monochrome, color or enhanced graphics display.
External Box:	The emulator boards can be installed in an external box with serial communication to the PC.
Processors supported:	8051, 8052, 8031, 8032, 80C51, 80C52, 80C31, 80C32, 8XC053/54, 83/80CL410, 83/80C451, 8XC528, 8XC550, 83/80C552, 80C562, 8XC575, 8XC592, 83/80C652, 83/87C751, 83/87C752, 83/80C851 (For more details, please refer to the following pages.) A switch from one microcontroller to another is made by changing the low cost POD.
File formats supported:	Intel HEX/OBJ/SYM/OMF, Avocet, Archimedes/IAR, Franklin/Keil, and many more.
Clock speed:	Allows operation up to 33 MHz in real-time.
Power supply:	The boards are powered from the PC-bus. The Emulator board requires 1.7A/5V and the Trace board 1.3A/5V.

nohau
CORPORATION
(408)866-1820

5.0 Ordering information

PHILIPS/SIGNETICS Microcontroller Support

EMULATOR UNITS

Includes software and ribbon cable. Must be connected to a POD to operate. Each step up in frequency rating covers all lower frequency steps. Includes DIP isolator if ordered with 40-pin POD. Emulator software has high-level debug and all options.

Order Number	Description
EMUL51-PC/E32	12 MHz Emulator, 32 kB (kiloByte) Emulation Memory.
EMUL51-PC/E32-16	16 MHz Emulator, 32 kB Emulation Memory.
EMUL51-PC/E128	12 MHz Emulator, 128 kB Emulation Memory.
EMUL51-PC/E128-16	16 MHz Emulator, 128 kB Emulation Memory.
EMUL51-PC/E128-20	20 MHz Emulator, 128 kB Emulation Memory.
EMUL51-PC/E128-24	24 MHz Emulator, 128 kB Emulation Memory.
EMUL51-PC/E128-30	30 MHz Emulator, 128 kB Emulation Memory.

TRACE BOARD OPTIONS

Optional second PC plug-in board. Emulator board contains no trace capability. Each frequency step covers all lower steps.

Order Number	Description
EMUL51-PC/TR4	12 MHz 4 kiloframe (4096 frames) Trace Buffer.
EMUL51-PC/TR4-16	16 MHz 4 k Trace Buffer.
EMUL51-PC/TR16	12 MHz 16 k Trace Buffer.
EMUL51-PC/TR16-16	16 MHz 16 k Trace Buffer.
EMUL51-PC/TR16-20	20 MHz 16 k Trace Buffer.
EMUL51-PC/TR16-24	24 MHz 16 k Trace Buffer.
EMUL51-PC/TR16-30	30 MHz 16 k Trace Buffer.

Advanced Trace Boards feature 32-bit timestamping with 16-bit prescaler, eight-level triggers, state and counter functions, search.

Order Number	Description
EMUL51-PC/ATR64-16	16 MHz 64 k Advanced Trace Buffer.
EMUL51-PC/ATR256-16	16 MHz 256 k Advanced Trace Buffer. Contact Nohau for availability.

BONDOUT PODS, 24-PIN, 40-PIN, 68-PIN, 84-PIN

Allow full emulation of internal, external and mixed modes of bus or port input/output. On-board POD crystal is 12 MHz on all PODs of any frequency specification. Each step up in frequency rating covers all lower frequency steps.

Order Number	Description
POD-C51B	12 MHz Bondout POD for 80C51, 8051, 87C51, 8751, 80C31, 8031, 80C/83C652, 83C654, 80C/83C662, 80C/83C851 single-chip or external mode.
POD-C51B-16	16 MHz Bondout POD for 80C/87C51-1, 80C/87C51, 80/8751, 80C31-1, 80C31, 8031, 80C/83C652, 83C654, 80C/83C662, 80C/83C851 single-chip or external mode.
POD-C51B-24	24 MHz POD for 80C/87C51-24, 80/8751, 80C31-24, 80C/83C652/654, 80C/83C662, 80C/83C851 single-chip or external mode. **Special Requirement:** Due to bondout chip timing, this POD requires a 30 MHz emulator board and trace board must be 30 MHz if used.
POD-CL410	POD for 83CL410, 83CL610. Target voltage range: 1.5–5.0V. Maximum frequency: 12 MHz at 5.0V. This POD is intended for emulating internal code. External bus operation is limited.
POD-C451B-PGA	12 MHz Bondout POD for 83C451, 87C451, 80C451 PLCC, single-chip or external mode. PGA from POD. Use optional adapter for PLCC target.
POD-C552B-PGA	12 MHz Bondout POD for 83C552, 87C552, 80C552 PLCC, single-chip or external mode. PGA from POD. Use optional adapter for PLCC target.
POD-C552B-PGA-16	16 MHz Bondout POD for 83C552, 87C552, 80C552 16 MHz, 12 MHz single-chip or external mode. PGA from POD. Use optional adapter for PLCC target.
POD-C652B	Order as POD-C51B.
POD-C751	12 MHz 83C751, 87C751, DIP POD. Includes EXT-DIP24.
POD-C751-16	16 MHz 83C751, 87C751, DIP POD. Includes EXT-DIP24.

nohau
CORPORATION
(408)866-1820

"HOOKS" MODE PODS, 28-PIN, 40-PIN, 42-PIN, 44-PIN, 68-PIN

Standard chip operated in special "hooks" emulation mode for single-chip or external modes. Some electrical characteristics are different from microcontroller's. On-board POD crystal is 12 MHz on all PODs of any frequency specification.

Order Number	Description
POD-C52	12 MHz POD for 80C31, 80C32, 80C51, 80C52, 87C51, 87C52.
POD-C52-16	16 MHz POD for 80C31, 80C32, 80C51, 80C52, 87C51, 87C52.
POD-C528	12 MHz POD for 80C528, 83C528, 87C528.
POD-C528-16	16 MHz POD for 80C528, 83C528, 87C528.
POD-C550-PGA	12 MHz POD for 80C550, 83C550, 87C550. PGA from POD. Use optional adapter for PLCC target.
POD-C592-PGA	12 MHz POD for 8XC592. 8XC592 chip may have to be supplied by user. PGA from POD. Use optional adapter for PLCC target.
POD-C592-PGA-16	16 MHz POD for 8XC592. 8XC592 chip may have to be supplied by user. PGA from POD. Use optional adapter for PLCC target.
POD-C752	12 MHz 83C752, 87C752 DIP POD. Includes EXT-DIP28.
POD-C752-16	16 MHz 83C752, 87C752 DIP POD. Includes EXT-DIP28.
POD-C054	12 MHz POD for 8XC053, 8XC054 (Micocontroller-for-Television-Video).
POD-C575	16 MHz POD for 8XC575.

POD BOARDS, 40 PIN EXTERNAL MODE

Port 2 is upper address bus only. Port 0 is address/data bus. P3.6 is WRITE, P3.7 is READ. On-board POD crystal is 12 MHz on all PODs of any frequency specification. Each step up in frequency rating covers all lower frequency steps.

Order Number	Description
POD-31	12 MHz 8031 POD.
POD-C31	12 MHz 80C31 POD.
POD-32	12 MHz 8032 POD.
POD-C32	12 MHz 80C32 POD.
POD-C652	12 MHz 80C652 POD.
POD-C31-1	16 MHz 80C31-1 POD.
POD-C32-16	16 MHz 80C32 POD.
POD-C31-20	20 MHz 80C31 POD.
POD-C31-24	24 MHz 80C31 POD.
POD-C31-30	30 MHz 80C31 POD.

-S version of the above boards allows P3.6, P3.7 to be used either as unidirectional I/O or write and read. (Example: POD-31-S; POD-C31-S-1; POD-C32-S-16.)

EXTERNAL MODE PODS, 64-PIN, 68-PIN

Port 2 is upper address bus only. Port 0 is address/data bus. P3.6 is WRITE, P3.7 is READ. On-board POD crystal is 12 MHz on all PODs of any frequency specification. Each step up in frequency rating covers all lower frequency steps.

Order Number	Description
POD-C451-DIP	12 MHz 80C451 DIP POD.
POD-C451-DIP-16	16 MHz 80C451 DIP POD.
POD-C451-PGA	12 MHz 80C451 PLCC POD. PGA from POD.
POD-C451-PGA-16	16 MHz 80C451 PLCC POD. PGA from POD.
POD-C552-PGA	12 MHz 80C552 POD. PGA from POD. Use optional adapter for PLCC target.

BOX OPTIONS

Box units are AC line-powered. Emulator software runs under DOS on PC and uses a COM port at 110 baud to 115 kilobaud. Serial cable included.

Order Number	Description
EMUL51-PC/BOX-S	Serial Box. Select emulator separately; trace optional. POD not included.
EMUL51-PC/BOX-CS	Serial Box with E128-16 Emulator and TR16-16 Trace. POD not included.
EMUL51-PC/BOX-CS-20	Serial Box with E128-20 Emulator and TR16-20 Trace. POD not included.
EMUL51-PC/BOX-CS-24	Serial Box with E128-24 Emulator and TR16-24 Trace. POD not included.
EMUL51-PC/BOX-CS-30	Serial Box with E128-30 Emulator and TR16-30 Trace. POD not included.

MISCELLANEOUS OPTIONS

Order Number	Description
EMUL51-PC/PRG	Universal Programmer (EPROM, 8751, 87C51, PAL).
EMUL51-PC/PRG751	Programmer for 87C751, 87C752.
EMUL51-PC/EXT-DIP24	Additional externder cable for 24 pin DIP.
EMUL51-PC/DIP24-ISO	24 pin DIP Isolator.
EMUL51-PC/DIP24-PLCC28	24 pin DIP to 28 pin PLCC.
EMUL51-PC/EXT-DIP24-PLCC28	Extender cable, 24 pin DIP to 28 pin PLCC.
EMUL51-PC/EXT-DIP28	Extender cable, 28 pin DIP.
EMUL51-PC/DIP28-PLCC28	28 pin DIP to 28 pin PLCC.
EMUL51-PC/DIP28-ISO	Additional 28 pin DIP Isolator.
EMUL51-PC/DIP28-DIP24-ADAP	28-pin (0.600-In) to 24-pin (0.300-In) adapter to plug POD-C752 into 8XC751 target. The user is responsible for port and register compatibility.
EMUL51-PC/EXT-DIP40	Extender cable for 40 pin DIP.
EMUL51-PC/DIP40-ISO	Additional 40 pin DIP Isolator.
EMUL51-PC/DIP40-PLCC44	40 pin DIP to 44 pin PLCC.
EMUL51-PC/DIP40-ONCE-DIP40	Clips over target microcontroller. Disables target micro to allow emulation without removing target chip. Works only on chips with on-chip emulation (ONCE) disable feature.
EMUL51-PC/DIP40-ONCE-PLCC44	Clips over target microcontroller. Disables target micro to allow emulation without removing target chip. Works only on chips with on-chip emulation (ONCE) disable feature.
EMUL51-PC/PGA44-PLCC44	PGA to PLCC adapter, 44 pin.
EMUL51-PC/PGA44-PLCC44-EL2	PGA to PLCC "elevator" or "tower" rigid 2-inch extender-adapter.
EMUL51-PC/EXT-DIP48	Extender cable for 48 pin DIP.
EMUL51-PC/DIP48-ISO	48 pin DIP Isolator.
EMUL51-PC/PGA68-PLCC68	PGA to PLCC adapter unit, 68 pin.
EMUL51-PC/PGA68-DIP84	PGA to DIP adapter unit for 451 PGA PODs to plug into DIP target.
EMUL51-PC/PGA68-ISO	68 pin PGA Isolator.
QILEXT-1	Extractor tool for PLCC parts.
EMUL51-PC/EZ	E-Z-Hook® wires for trace.
	E-Z-Hook is a registered trademark of Tektest, Inc.
EMUL51-PC/CBL10-S	10 foot substitute for 5 foot POD cable (not for bondout PODs).
EMUL51-PC/CBL10-A	Additional 10 foot POD cable (not for bondout PODs)
EMUL51-PC/CBL5-A	Replacement 5 foot POD cable

BANKSWITCH EMULATOR BOARDS

User selectable as two banks of 64 kBytes, or as three switchable banks in upper half (8000 – FFFF) with lower half (0000 – 7FFF) not switchable. MOVX read-write data memory is only separate from code if data is mapped to target.

Order Number	Description
EMUL51-PC/E128-BSW	12 MHz Bankswitch Emulator, 128 kB Emulation Memory. Requires Bankswitch POD.
EMUL51-PC/E128-BSW-16	16 MHz Bankswitch Emulator, 128 kB Emulation Memory. Requires Bankswitch POD.
-BSW option to any POD	
Examples:	
POD-31-BSW	12 MHz 8031 Bankswitch POD.
POD-C31-BSW-1	16 MHz 80C31-1 Bankswitch POD.
POD-31-S-BSW	12 MHz 8031 -S option Bankswitch POD.

nohau
CORPORATION
(408)866-1820

SOFTWARE PACKAGES

Order Number	Description
NOHAU Corporation	
SIMUL51-PC	Nohau 8031/8051 Simulator. Same interface as EMUL51-PC.
Software subscription	EMUL51-PC emulator software update service, one year.
Archimedes Software, Inc.	
ARCHM/C-8051PC	Archimedes C-8051PC V4 C-Compiler & Assembler.
ARCHM/SIM-8051PC	Archimedes 8051 Simulator/Debugger (SimCASE Sim-8051PC) for CC51, PL/M-51 & ASM.
	Archimedes is a trademark of Archimedes Software, Inc.
Avocet Systems, Inc.	
EMUL51-PC/AVA51	Avocet Systems AvCase51 8051 Assembler
	Avocet is a registered trademark of Avocet Systems, Inc.
BSO/Tasking	
BSOTSK/C51PKG	BSO/Tasking 8051 Family C-Compiler and Assembler Package.
BSOTSK/PLM51PKG	BSO/Tasking 8051 Family PL/M Compiler and Assember Package.
Franklin Software, Inc.	
FRANKLIN/4010	Franklin 8051 Macro Assembler with Linker, Librarian, Object-to-Hex utility.
FRANKLIN/5010	Franklin 8051 V2 Medium-High Performance C-Compiler & Assembler.
FRANKLIN/5020	Franklin V3 Very High Performance 8051 Compiler & Assembler.
FRANKLIN/7010	Franklin V3 Simulator/Debugger, ASCII Interface.
FRANKLIN/7020	Franklin V4 Simulator/Debugger, Windowed Interface.
FRANKLIN/8210	Franklin 5010 C-Compiler, 4010 Assember and 7010 Simulator/Debugger Package.
FRANKLIN/8220	Franklin Developers Kit with 5020 Compiler, 4010 Assembler and 7020 Simulator/Debugger.
	Franklin is a trademark of Franklin Software, Inc.
Intermetrics Microsystems Software, Inc./Whitesmiths, Ltd.	
EMUL51-PC/IWCC51S	Intermetrics Whitesmiths C-Compiler and Assembler, Standard.
EMUL51-PC/IWCC51X	Intermetrics Whitesmiths C-Compiler and Assembler, Extended.
EMUL51-PC/IWCXDBE51X	Intermetrics Whitesmiths C Source-Level Debugger/Emulator Version (Interface for EMUL51-PC).
EMUL51-PC/IWSIM51	Intermetrics Whitesmiths C Source-Level Debugger/Simulator Version.
	Whitesmiths and CXDB are registered trademarks of Intermetrics, Inc.

The EMUL51-PC Emulator, Trace, POD, and Box hardware is sold with a one-year warranty. The EMUL51-PC Emulation software is sold with no warranty, but upgrades will be distributed to all customers up to one year from the date of purchase. Nohau Corporation makes no warranties, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. In no event will Nohau Corporation be liable for consequential damages. The SIMUL51-PC Simulator software includes updates for a period of one year. Third-party software and programmers sold by Nohau carry manufacturers' warranties.

nohau
CORPORATION
(408)866-1820

CHOOSING PODS for the EMUL51-PC

People sometimes ask: "How do I choose which POD to use for the 8051-family chip I want to emulate?"

Which POD you pick for your application depends on several things. What micro you are using, the frequency, the mode you are using it in, your target configuration, and price all play a part. The **EMUL51-PC ORDER INFORMATION** can help you choose among the POD types available from Nohau.

Nohau has several categories of PODs: (1) External-mode PODs; (2) Bondout PODs; and (3) "Hooks"-mode PODs.

1. External-Mode PODs

External or microprocessor-mode PODs have several clear advantages. They are the cheapest type of POD. They have easily-replaceable standard microcontrollers. Your program runs on the real production part, with most of the lines directly connected to your target. You can use them where your microcontroller has external program or external read-write memory. With this type of POD, Port 2 is used only for the upper address bus and not as port input-output (I/O). Port 0 is used as a multiplexed address and data bus. In most of these PODs, P3.6 can only be used as the write line and P3.7 as only the read line. The **POD-31** is a typical external POD.

Your target system shares Port 0 and Port 2 with the emulator. The emulator uses Port 0 to run its monitor code when it is not executing your code in real-time. When you are not running real-time emulation, the POD holds the Program Store ENable line (PSEN/) high so that your external read-only memory (ROM) is not enabled. It also holds the ReaD/ line and WRite/ lines high so that none of your external read-write random access memory (RAM) or I/O is enabled.

But in the monitor (not emulating) mode, the address lines of both Port 2 and Port 0 are active and can output any address in the 64 kByte address range. It is up to your target system to prevent enabling any device unless the PSEN/ or the RD/ or the WR/ line goes low.

If you are emulating a ROM-less part, like the 8031, you can use the **POD-31** type of POD. You also might use this POD for some internal ROM applications. You could use it for designs that have all of Port 2 and Port 0 used as external buses. This is true even though the program can be executed from on-chip 8051 program memory in your final product. This is a case where your target schematic, rather than the device you are using lets you use this type of POD. So though the part may ultimately be an 8051 or 8751, you can use a **POD-31** because you are using the ports as buses.

If you are emulating the 80C31, 8032, 80C32 or 80C652, you can get external-mode PODs for these microcontrollers. With the **POD-31**, you can emulate all those parts in external mode at up to 12 MHz. You can get the POD with the correct micro installed, such as a **POD-32**. Or you can change among the types by changing the microcontroller component in the POD yourself.

16 MHz, 20 MHz, 24 MHz, 30 MHz and 33 MHz PODs are available for 40-pin parts. If you use a **POD-C31-1** with a 16 MHz rated emulator, you can support such parts as the 80C31-1, 80C32-1 and 80C652-1. Any higher-frequency POD and emulator set can support all the lower frequencies.

For emulating the 80C451, 80C452 or 80C552, other external mode PODs are available. PODs for dual inline package (DIP) parts have a DIP plug coming out of the bottom of the POD. PODs for plastic leaded chip carrier (PLCC) parts have a pin-grid array (PGA) plug coming out from the bottom of the POD. To plug a PGA POD into a PLCC socket, you can get an optional adapter. The PGA68-PLCC68 is a typical adapter. If your target board has feed-through holes and you solder a PGA socket into it, you don't need an adapter. If your target board already has a PGA socket, you don't need an adapter.

POD-31-S: All the above external PODs use P3.6 as WR/ and P3.7 as RD/. But there is a 40-pin external mode POD version that lets you use these two lines as I/O. The basic board is the **POD-31-S**. This special -S option lets you select whether the two lines are to be WR/ and RD/ or else I/O. They can be unidirectional port lines (input-input, input-output, output-input, or output-output). You also can get this POD in the variations of 16 MHz and 20 MHz and any of the 40-pin part variations listed above.

2. Bondout PODs

Bondout PODs use special microcontrollers that the semiconductor manufacturers designed to perform all the functions of the part. They also have additional lines "bonded out" from the chip that allow control for emulation. PODs with these special chips cost more than PODs with widely available commercial chips, but give you a greater flexibility in how you use the ports. These PODs allow you to use any combination of internal or external mode of the chip, and respond to the state of the External Access (EA/) pin.

Nohau has bondout PODs for many different microcontrollers.

You can emulate the 80/80C51, 87/87C51, 80/80C31, 80C/83C/87C652, 83C/87C654, 80C/83C662 and 80C/83C851 with the **POD-C51B**. A 16 MHz version is available.

To emulate the 83C451, 87C451 and 80C451 in the PLCC package, you can use the **POD-C451B-PGA**. See the previous discussion about PGAs.

For emulating the 80C/83C/87C552 and 80C/83C/87C562 you can use the **POD-C552B-PGA**. It also is available in a 16 MHz version.

For emulating the 83C751 and 87C751, use the **POD-C751**. It has a 24 pin DIP plug on the bottom of the POD. You also can get an adapter to plug into 28 pin PLCC sockets. A 16 MHz version is available.

For emulating the 8XCL410 and 8XCL51, use the **POD-CL410**.

You should be aware of one small problem that most 8051 bondout parts have with serial communication. Specifically, if your program has written to SBUF, but the TI flag has not been set, and you then break emulation, the TI flag will never be set after you resume emulation.

Bondout PODs give you the best of both worlds. They let you mix how you use the ports. So you can emulate the single-chip mode using the ports as I/O. Or you can emulate external bus mode. Or you can emulate a mix of both modes.

3. "Hooks"-Mode PODs

The newest type of POD Nohau offers uses a different technique for emulating. Certain chips can be put into a proprietary "hooks" mode that multiplexes the port information in and out of the part. The chip can still put out address and data. Most have 16 MHz versions available. This group includes these PODs:

The **POD-C752** is for emulating the 83C752 and 87C752. It has a 28 pin DIP plug on the bottom of the POD. You also can get an adapter to plug into 28 pin PLCC sockets.

The **POD-C52** can emulate the 80/80C/87/87C51, 80/80C/87/87C52, 80/80C31 and 80/80C32. You can use this POD when you need to emulate the 8052-type parts in single-chip mode when you are using Timer 2.

The **POD-C528** can emulate the 80/80C/87/87C528.

The **POD-C592-PGA** can emulate the 8XC592. See the previous discussion about PGAs.

The **POD-C550-PGA** can emulate the 80/80C/87/87C550. It has a 44-pin PGA plug. You also can get an adapter to plug into 44-pin PLCC sockets.

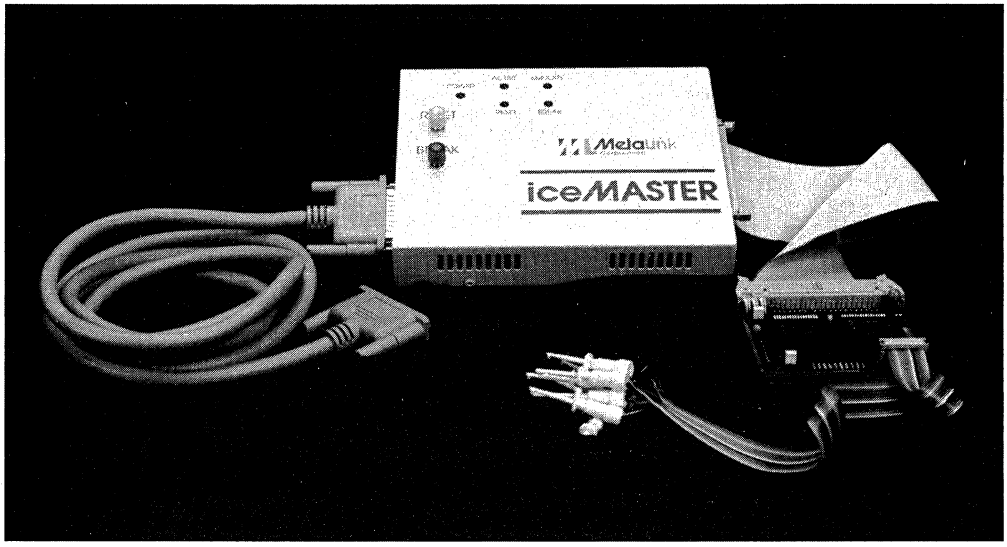
The **POD-C054** can emulate the 8XC053/054 "MTV" chip.

With the "Hooks"-Mode PODs, you use jumpers to select whether the code is to be fetched from the emulator RAM or from external ROM. You also select whether the read-write memory is in the emulator or on your target board. The ports on these PODs may have slightly different electrical characteristics than the microcontroller. Specifically, some output signals can appear up to three clock cycles later than on the actual part. Also, some ports have greater current sink or sink and source capacity or slightly higher impedance than the actual part. For almost all applications, these differences will have no detrimental effects.

You can use any size **EMUL51-PC** emulator and optional trace board with any POD you choose in Nohau's **EMUL51-PC** family. Choose an emulator and trace board with frequencies as fast or faster than the frequency of your fastest POD. If you need more information about which POD would best fit your application, please contact Nohau.

MetaLink® Corporation

Looking for EMULATION PRODUCTIVITY? Look through our WINDOWS!



iceMASTER™

IN-CIRCUIT EMULATORS From MetaLink For 8051

- **iceMASTER delivers** productivity: easy to learn, easy to use, and fast!
- **Flexibility** will acquire new meaning when you experience the **iceMASTER** user interface. You can **completely configure** the windows for size, position, content, and color.
- **iceMASTER** is **FAST!** The 115.2K baud serial link keeps typical download times to under 3 seconds using a standard COMM port!
- **iceMASTER** is **convenient!** It connects easily to your PC, requires no disassembly, nor does it take up any expansion slots. It works on any PC (DOS or OS/2), Micro Channel, ISA, or EISA. Even Laptops!
- **iceMASTER**-8051 supports numerous different family derivatives.
- **iceMASTER** is **RISK FREE!** If you are not completely satisfied during a trial period, you can return the emulator for a full refund!
- To receive your **FREE DEMO DISK**, CALL **(800) METAICE** TODAY!!!
  (800) 638-2423

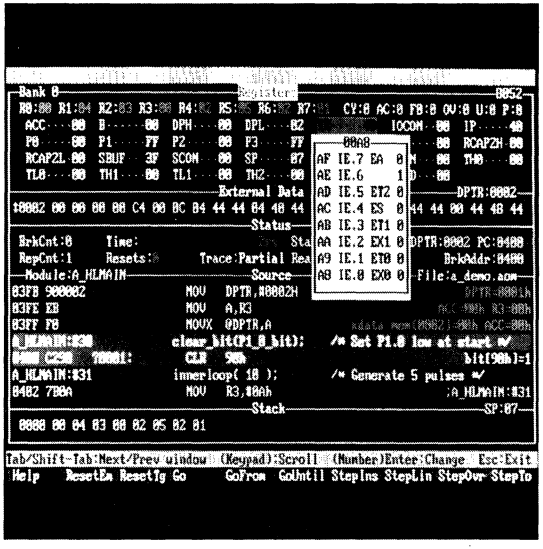
©1991 MetaLink Corporation

FLEXIBLE, EASY-TO-USE INTERFACE

iceMASTER has an advanced color windowed user interface that emphasizes ease of use. Each window can be sized, moved, highlighted, color–controlled, added, or removed completely. iceMASTER provides pull–down and pop–up menus, function keys, and context–sensitive help. The contents of any memory space may be perused and altered directly from the appropriate window using the keyboard or a mouse.

You have immediate access to the hypertext / hyperlinked, context–sensitive, on–line help system which explains clearly what your options are (at any detail level you choose), keeping you productive. There is even a HELP–FOR–HELP feature. Whether you are beginning your first emulation project, or are a veteran designer searching for the fastest possible debugging method, you will appreciate the EASE–OF–USE features designed into iceMASTER.

Novices can navigate smoothly through a debugging session by accessing the commands and menus as standard pull–downs. Experienced designers can instantaneously pop–up the menu of choice using redefinable hot keys.



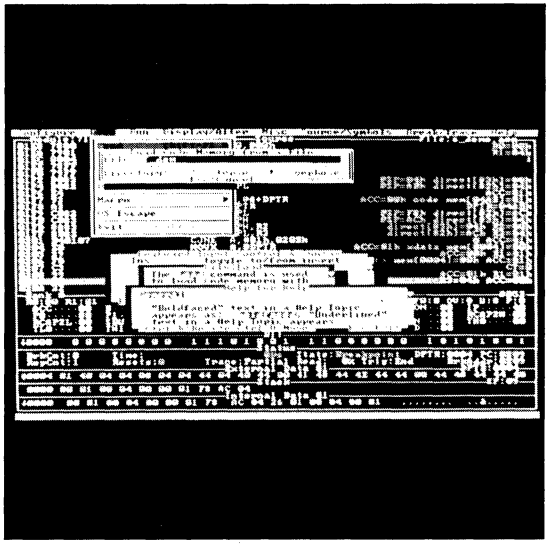
THE COMPLETE DEBUGGER

The 4K–frame trace buffer captures data in real–time. Trace information consists of: address and data bus values, and user–selectable probe clips. You can view the trace buffer data through several display filters: raw hex, disassembled instructions, instructions mixed with HLL source statements, or HLL source only; you can display the probe clip bit values in binary, hex, or digital waveform formats.

You can trigger the trace to begin capturing data on all instructions leading up to a breakpoint, around (before and after) a breakpoint, or following a breakpoint. Capture filtering allows you to focus attention only on areas of interest, eliminating clutter.

To further speed your design process, an integrated search mechanism allows you to locate any label, HLL source line number, or address in the trace buffer, in either the backward or forward direction. The days of manually scanning or post–processing a large buffer of trace data are gone!

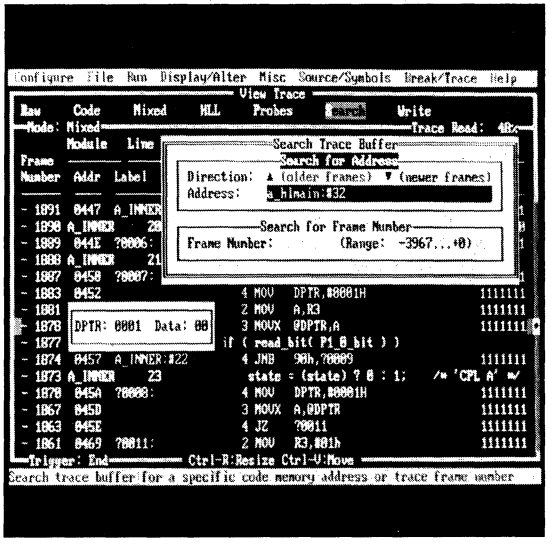
If you WRITE your program in a High Level Language (HLL), you should be able to DEBUG it that way — iceMASTER lets you do just that!



DECREASED DEVELOPMENT TIME

Your iceMASTER source window accelerates the debugging process with Dynamically Annotated Code. When single–stepping, instruction execution information is displayed and RETAINED next to each instruction. You can clearly see the data behind your program's flow, including contents of all accessed (read or write) memory locations and registers, as well as flow–of–control direction change markers. A moving color bar indicates the current position in the program as it executes.

High Level Language (HLL) source statements and symbolic disassembly information are also displayed when you disassemble the program or view the trace buffer.

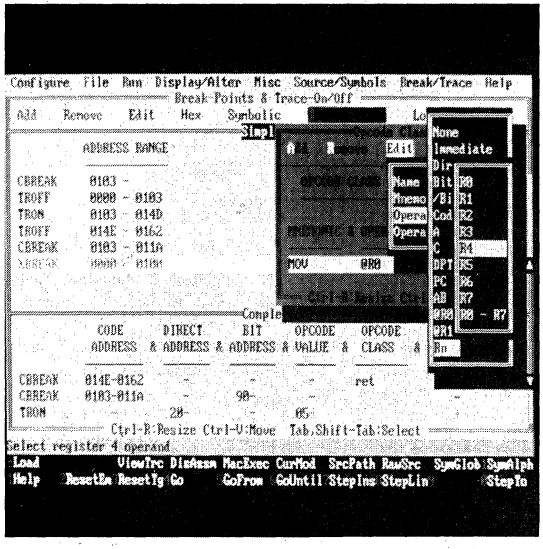


IMPROVING THE QUALITY OF YOUR PRODUCT: FINDING THE HOT (OR NOT SO HOT) SPOTS

iceMASTER provides a PERFORMANCE ANALYZER that is the absolute best in the industry for tuning and testing your code. It is very flexible and far more accurate than most other emulators, with a resolution of less than six microseconds. You monitor the time spent executing specific portions of your program to find "hot spots" or "dead code."

You can define and analyze up to 15 memory areas based on code address, module, line number or label ranges. If you choose, each of these 15 areas can consist of non–contiguous, logically related subranges, (e.g. groupings of related functions which are not necessarily contiguous in memory).

You can view results dynamically during emulation or later for a more detailed analysis. You can toggle the display from bar graph format to actual frequency counts, and, of course, you can filter the level of detail in the displayed results to include raw data, code labels, and/or HLL source statements.



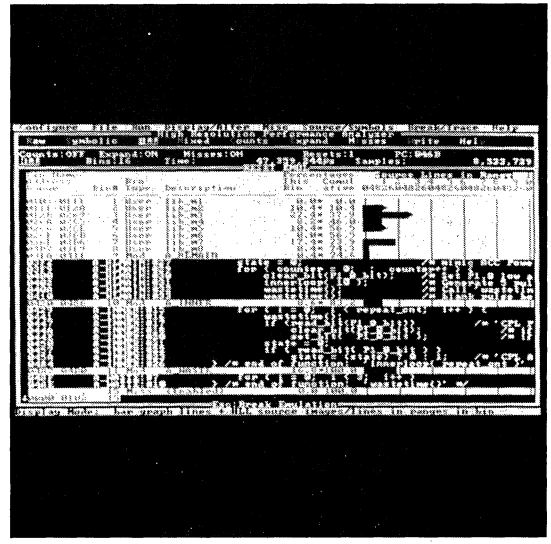
SMALL AND POWERFUL

Some companies design emulators that use one or two full size PC expansion board slots. These companies take it for granted that you have the "right" kind of PC, that you don't need the expansion slots for something else, and that you don't mind taking apart your computer to install their emulator!

At MetaLink, we don't take our customers for granted.

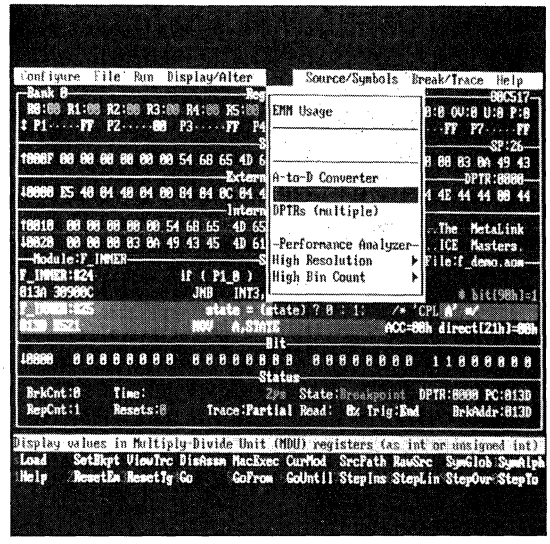
iceMASTER is the culmination of over 6 years of focused engineering by MetaLink to bring advanced semiconductor technologies to emulator design. Using state–of–the–art PALs (PEELs), LCAs, VLSI memories and microprocessors, MetaLink designed world–class emulation capability into the smallest emulator footprint in the industry, combining powerful and complex emulation features with all–around ease–of–use.

The high speed serial link connects easily to the back of your computer with a standard RS–232 cable. The emulator, about the size of a VCR tape, fits neatly in crowded work spaces and is easy to move around on the bench or to other computers. PORTABLE PRODUCTIVITY!



FINDING THAT BUG

You have access to as many as 128K break and 64K trace triggers. These triggers can be enabled, disabled, set, or cleared. They can be simple triggers, based on code or external data addresses or address ranges. They can also be complex triggers, based on code address, direct address, bit address, opcode value, opcode class or immediate operand. Complex triggers can be ANDed and ORed together. Finding that elusive bug is now much easier!



MetaLink is DEDICATED

MetaLink specializes in enhancing the emulation technology required by EMBEDDED SYSTEMS DESIGNERS. MetaLink has consistently led the industry in emulation technology: the first PC-based 8051 emulator; the first 8052 emulator; the first to offer support for all the unique features of 8051 family components, such as watchdog-timer, idle mode, power-down mode, and A/D. Currently, other 8051 emulator suppliers must license MetaLink's patented technology to support certain devices.

MetaLink is a full-service emulation company. We support our customers over the long term with services such as repair, discounted upgrades, rental units, trial purchase periods, and free technical support for applications problems. A network of world-wide sales and service representatives is augmented by a well-trained telemarketing staff at headquarters.

SPECIFICATIONS AND OPTIONS

The iceMASTER series of in-circuit emulators was designed with options to fit most performance requirements and budgets. Customers may select between different models and options. Below are the detailed specifications and the configuration options.
Note: Certain specifications and options are specific to a particular chip family.

EMULATOR SYSTEM REQUIREMENTS
- Basic Emulator System Model 200 or Model 400
- Interchangeable Probe Card
- +5 Volt 1.5 Ampere Power Source
- 128K Memory Expansion

MODELS
- 200 Emulator
- 400 Emulator with:
  - 4K Trace Buffer
  - 2 Performance Analyzers
  - Full Watchdog Timer Support

FILE FORMATS
- MetaLink, Archimedes, Avocet, BSO, Franklin, IAR Systems, Intel OMF,Intermetrics, Keil, MCC, Microtec Research, Tasking, Whitesmith, and Intel HEX.

MACRO
- Repetitive routines
- User-created and callable

MEMORY OPERATIONS
- Program Memory:
  - Single line Assembler
  - Disassemble
  - Disassemble to file
  - View/Change
  - Mapping
- Data/Code Memory:
  - Dump
  - Dump to file
  - Fill
  - Move
  - Change
  - Compare
- Registers/SFRs/Bit Memory:
  - Examine/Modify
- Program Variables:
  - Examine/Modify

OPERATING CHARACTERISTICS
- Electrically Transparent
- Operationally Transparent

USER INTERFACE
- Keyboard or Mouse Control
- Pull-down & Pop-up menus
- Main Screen Windows:
  - Registers/SFRs/PSW bits
  - Bit Memory
  - Stack
  - Up to 5 Internal Data Memory
  - Up to 5 External Data Memory
  - Up to 5 Code Memory
  - Source Program
  - Watch
  - System Status
- User Window Controls:
  - Selectable (On/Off)
  - Movable
  - Resizable
  - Scrollable
  - Color selection
  - Highlighting
- Function/Hot Key Access:
  - User-assignable

EMULATION CONTROLS
- Reset from Emulator
- Reset from Target
- Reset Processor
- Go
- Go From
- Go Until
- Slow Motion
- Step
- Step Line
- Step Over
- Step To
- Repetition Counter

DEVICE-UNIQUE SUPPORT
- A-to-D Converter
- PROM Programmer

PERFORMANCE ANALYZER
- Real-time program profiling
- 5.4 usec. sampling period
- 7 year duration
- Display options:
  - Bar Graph
  - Frequency Count
- Display Modes:
  - Raw
  - Symbolic
  - HLL Source lines
  - Mixed
- Up to 15 Bin capacity:
  - Multiple ranges per Bin
- User-controlled Bin set-up:
  - By Address
  - By Symbol
  - By Module
  - By Line Number
  - Automatic

TRACE
- Trace Triggers:
  - Start
  - Center
  - End
  - Variable
- 4K-Frame Trace Buffer
- Trace Contents:
  - Address
  - Data
  - External Clips
- Trace Display Modes:
  - Raw Hex
  - Symbolic
  - HLL Source
  - Mixed
  - Binary (Clips)
  - Digital waveform (Clips)
- Trace Buffer Operations:
  - Write buffer to a File
  - Search Trace Buffer

HELP
- On-line
- Context sensitive
- Hypertext/ Hyperlinked

SOURCE/SYMBOL SUPPORT
- C and PL/M
- Source-level debug

HLL Structure/Content Display:
- Modules
- Scopes
- Line Numbers
- Program Variables

ELECTRICAL SPECIFICATIONS
- Input Power (maximum):
  - 1.5 A @ +5 VDC +/-5%

MECHANICAL SPECIFICATIONS
Emulator Dimensions:
1.0" x 7.0" x 5.5"
2.5cm x 17.8cm x 14cm
Probe Card cable length:
14.0" 35.6cm
Emulator weight:
2.0lbs 0.9kg

WARRANTY
One (1) year limited warranty, parts and labor, for registered users.

Specific Device Support

iceMaster Family	8051 Std.	8051 Opt.
EMULATION MEMORY:	32K	128K
Program	16K	64K
External Data	16K	64K
MAPPING RESOLUTION:		
Program(Bytes/bloc)	16	
External Data(Bytes/bloc)	16	
REAL TIME:	DC-24MHz	
BREAKPOINTS:	128K	
TRACE ON:	64K	
TRACE OFF:	64K	
TRIGGER CONDITIONS:		
PC address & range	X	
Opcode Value	X	
Opcode Class	X	
SFRs/Registers	X	
Direct byte address & range	X	
Direct bit address & range	X	
Immediate operand value	X	
Read/Write to bit address	X	
Register addr. modes	X	
Read/write to Register addr	X	
Logical AND/OR of any of the above	X	
Ext. Data Address & range	X	
Break Count overflow	X	
External Input	X	
OPERATING MODES:		
ROM	X	
ROMless	X	

DEVICES SUPPORTED

Interchangeable Probe Cards are used with the appropriate emulator to support the functional derivatives of each microcontroller as well as the full range of NMOS, CMOS, EPROM, and OTP technology variations. Since MetaLink is constantly adding to its list of supported devices, please contact MetaLink for information on any device not listed.

iceMASTER-8051 supports: 8031, 8051, 8032, 8052, 80C451, 83C451, 80C552, 83C552, 80C652, 83C652, 80C562, 83C562, 83C053, 83C054, 80C550, 83C550, 83C751, 83C752, 80C851, 83C654, 80CL410, 83CL410, 80C528, 83C528, 83C524, 80CL51.

Product names are used for purposes of identification only and may be trademarks or registered trademarks of their respective companies.

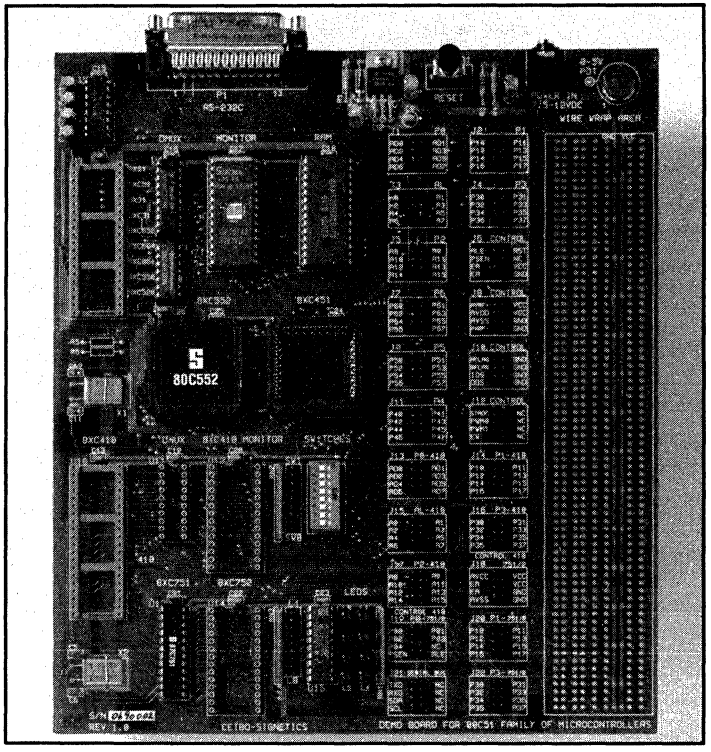
To order your iceMASTER emulator or to arrange a demonstration of iceMASTER, call:1-(800)METAICE or 1-(800)638-2423.
Rental plans are available.
Ask for details on this opportunity.

MetaLink Corporation
325 E. Elliot Road
Chandler, Az 85225
Phone: (602) 926-0797
Fax: (602) 926-1198
Telex: 499 8050 MTLNK

DB-51
CEIBO Development Board

DB-51 is a high-performance system design board dedicated to the Signetics 80C51 family of microcontrollers. It provides an easy-to-use flexible instrument which enables the user to build a primary prototype, analyze and debug it, make changes, and continue debugging. And you can improve your design decisions by using the DB-51 to check and test the advantages of several different microcontrollers. The DB-51 is also a great training and tutorial aid for becoming familiar with designs using the 80C51 architecture. Note that the DB-51 is not intended to replace a full emulator system in complex microcontroller designs.

- Supports most of the Signetics 80C51 derivative microcontrollers
- Serially linked to IBM PC or compatible hosts
- 32K of user code memory
- Software breakpoints
- Examine and alter chip registers, RAM, and ports
- Symbolic debugger compatible with linker object files
- Upload and download of object and hex files
- Special wire-wrap area for prototyping
- User's manual with examples and applications designed to familiarize the user with 8XC51 architecture and programming as well as the use of the DB-51 itself

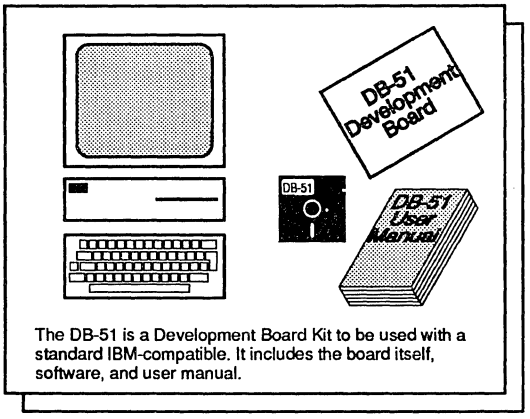


Signetics

Philips Semiconductors

PHILIPS

SPECIFICATIONS



The DB-51 is a Development Board Kit to be used with a standard IBM-compatible. It includes the board itself, software, and user manual.

System Memory

DB-51 provides 32K of user code memory. This RAM memory permits downloading and modifying of users' programs.

Breakpoints

Breakpoints allow real-time program execution until an opcode is executed at a specified address.

Symbolic Debugger

DB-51 allows symbolic debugging of assembler or high-level languages. The symbolic debugger uses symbols contained in the absolute file generated by the most commonly used relocator and linker programs.

Supported Microcontrollers

8X31/51, 8X32/52, 8XC31/51, 8XC32/52, 8XC652, 8XC654, 8XC851, 8XC550, 8XC552, 8XC562, 8XC451, 8XC528, and others with external memory addressing and a UART are fully supported. 8XCL410, 8XC751 and 8XC752 have very limited support.

Limitations

"Fully supported" microcontrollers are self-debugging on the DB-51. Thus, some of the chip resources are used by the board: the monitor program uses the bottom 32K of program memory; chips are always operated in the external memory mode; the UART is used to communicate to the PC and is thus not normally available to the user program; interrupt response is slowed slightly by re-vectoring from the monitor program to the user program; use of watchdog timers and power-down and idle modes of operation are limited due to interaction with the monitor program.

"Limited support" microcontrollers do not have on-chip UARTs and most do not support external program memory. Thus, download of programs to these parts is not supported on the DB-51. The 87C751 provided with the board is pre-programmed with a "micro" monitor program and some predefined experiments described in the user manual. Also, these parts use the I^2C bus to communicate to the PC, limiting the use of I^2C for other purposes.

User Software

The board is provided with a very easy-to-use menu-driven software program as well as command oriented user interface software. On-line assembler and disassembler are provided together with upload and download capabilities of hexadecimal and object files.

Command Set

ASM – BIT – BYTE – BREAKPOINT [enable, disable, reset] – CHIP [type] – CLS – CODE – DATA – DASM – DEFAULT – DIR – EVALUATE – EXIT – GO [from, till] – HALT – HELP – HISTORY – LINES – LIST [file] – LOAD [code, symbols] – LOCALS – MODULES – PORTS – PROCEDURES – PUBLICS – RBIT – RBYTE – REGISTERS – RESET – SAVE – SOUND – STATUS – STEP [n] – TIME

Host Characteristics

IBM PC/XT/AT or compatible system with 512 kbyte of RAM, one floppy disk drive, one RS-232 interface board for the PC and cable, PC-DOS 2.0 or later.

Input Power

7.5 VDC to 12.0 VDC (9 VDC wall transformer supplied).

Mechanical Dimensions

20 cm x 25 cm

Items Supplied as Standard

DB-51 board, 80C552 and 87C751 microcontrollers, monitor EPROM, power supply, RS-232 cable.
User software including symbolic debugger, on-line assembler and disassembler.
User's manual and operating instructions.

ORDERING INFORMATION

Order part number P8051DBSD from your local Signetics Distributor.

For more information, contact us today: (800)227-1817, ext. 737.

Stand-alone debug station for 80C51/8051-based systems

SDS 8051

THREE CONFIGURATIONS TO SUIT YOUR 8051 PROJECTS

For efficient, accurate software and hardware development and integration, there is no substitute for fully transparent, real-time emulation. But development must be cost-effective, and the key to that is to have *one* emulation unit which can fit exactly into *all* your development projects irrespective of their scope and stage of development.

If you are developing with the 8051 family, such a unit is the SDS 8051.* Three configurations suit all your projects without requiring modifications or extras:

Stand-Alone Operation

A VDU Terminal is all that is needed to integrate and debug with the SDS 8051. And because you can do basic emulation without putting demands on computer resources, the smallest user as well as the big development team can take advantage of the SDS 8051.

Connection to a Host Computer

Most development tasks will, however, need the power of a host computer or Microcomputer Development System, and Philips' SDS (Stand-alone Debugging Station) can work with both. The process of writing the software and designing the hardware can be done on the computer, with an SDS used to debug and integrate the software with the hardware. SDS 8051 supports a wide range of hosts. It has its own in-line assembler in firmware, but you may also use the separate MS-DOS cross-assembler.

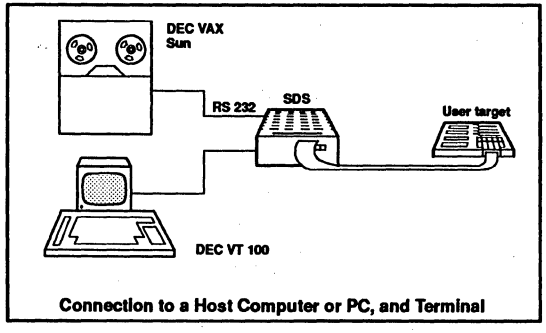
Connection to a PC

The SDS 8051 operates with an IBM PC or XT, or compatible, e.g., Philips P3100 series, so you can use it in your usual work environment. The SDS can be operated from a PC using readily available terminal emulation software, or the XRAY™51 symbolic debugging package, the latter allowing you to use your own source labels on screen in SDS displays and commands, and having a DOS toggle switch.

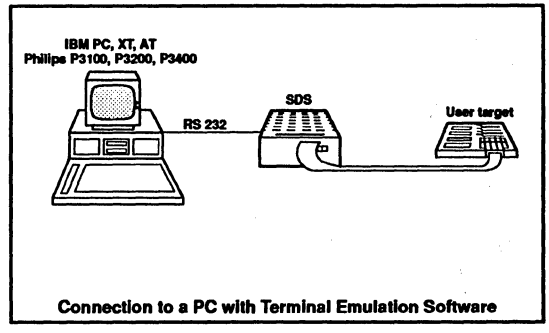
If you work in a development team, shared access to files and programs is of course essential to maximize productivity. Programs can be downloaded from any host computer to the SDS memory, using a PC or terminal as a workstation to operate the SDS.

FEATURES

- Real-time, fully transparent emulation
- Works stand-alone from a VDU terminal, PC or other computer
- Full hardware emulation using a dedicated probe—operates according to the exact specifications of the target microcontroller, including maximum speed
- Interfacing signals to external equipment
- Full assembly-level debugging and HLL debugging
- Single-step and breakpoint facilities
- Large trace memory with hardware qualifiers



Connection to a Host Computer or PC, and Terminal



Connection to a PC with Terminal Emulation Software

*SDS 8051: Generic name for Philips Stand-alone Debugging Station for the 8051 family of microcontrollers; for type numbers, see Ordering Information. The 8051 family includes the 80C51/31, 87C51, 80CL51, 8XC053/054/451/528/550/552/562/592/652/654/751/752/851, the 8XCL410/710 and the 8051/31/52/32. Debugging stations are also available for Philips 8400 microcontroller family and the 5010/5011 digital signal processors.

Stand-alone debug station for 80C51/8051-based systems

SDS 8051

UNEQUALLED EMULATION

SDS 8051 provides complete, high-quality emulation for development with the 8051 family of microcontrollers. Standard features include:

Fully Transparent Real-Time Emulation

You can develop on the SDS 8051 with the chip running in real time—no stretched clock cycles or wait states to disrupt system timing in the prototype. And emulation is fully transparent—since no resources from the 8051 memory, I/O or interrupt space are used for monitoring emulation, all are available to the target and user program.

In-Circuit Emulation or Simulation

The SDS 8051 is versatile—you can tailor operation to suit the state of the prototype. If no prototype hardware is available, the SDS may be used for simulation, so the software can still be tested. The emulation mode runs the program in the prototype so far as it has been developed, with resources transferred in stages to the prototype. Hardware, software and their integration are fully tested.

Full Real-Time Hardware Emulation and Breakpoint Setting

The SDS 8051 has an 80C51 (or derivative) bond-out chip in the emulation probe. This, and only this, can ensure that hardware emulation is truly real-time, and it also allows you to set hardware breakpoints. These can be set on any combination of addresses, register values or branch instructions, allowing you to investigate program flow in detail and to debug very quickly. When a breakpoint condition is met, the entire CPU is frozen and, besides the full interrupt status, the status of all timers (frozen by a special bond-out chip feature) is displayed.

Alterable Memory and Processor Registers

For really quick fault-finding, the SDS allows you to alter, and to display, memory and CPU register values as required, allowing parts of the program to be repeatedly tested using convenient values.

In addition, the SDS has an emulator-resident in-line assembler which is particularly useful when changing your program—there being no need for repeated up-loading and down-loading.

Disassembly

Of course, there is no need to remember binary code references when developing software with the SDS 8051—instructions are entered in assembly language. In addition, on-board memory can be disassembled into the originally programmed instruction mnemonics.

Trace Memory

SDS 8051 has a 2048-line trace memory for quick checking of the program flow. The display shows the address, opcode/operand, disassembly, instruction status (such as RD OPC, WR) and the contents of microcontroller ports and/or eight user test clips.

Hardware Qualifier Bits

The trace memory has qualifier bits for selecting the information to be captured in the trace memory. These bits enable cycles to be selected, for example, capture only on fetches from emulation memory, or on interrupt acknowledge cycles. Selecting the information before capture overcomes the drawback of software qualifiers which select the information afterwards, relying on luck for it to be in the trace memory!

Debugger Commands

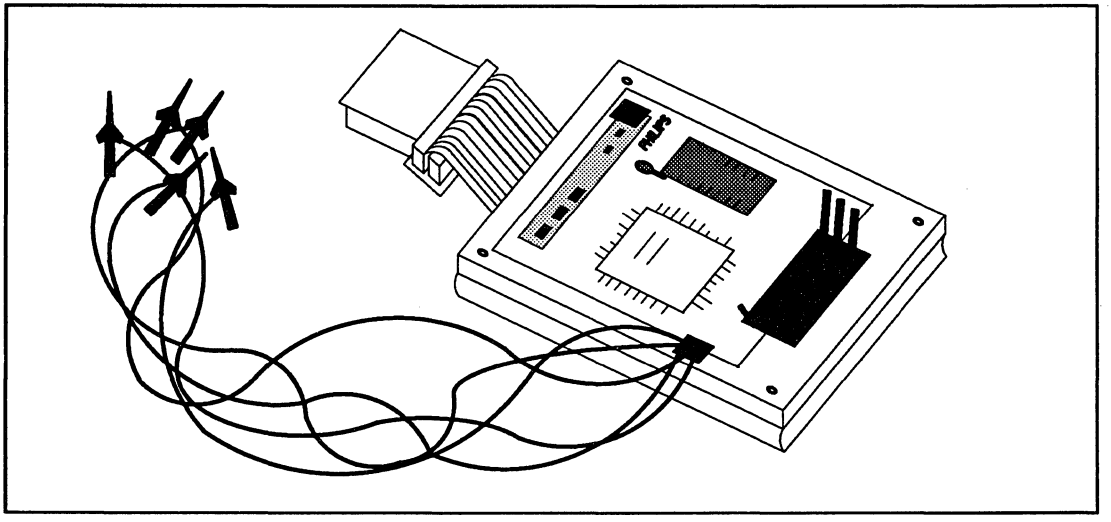
The microprocessor development command language employed by the SDS 8051 has been designed for ease of use. Commands form a subset of, and are similar to, the widely used ICE™ language.

When the XRAY51 symbolic debugging package is used, the commands are the XRAY commands. These and the SDS commands are listed below (at the end of this description).

Software Packages

The XRAY51 high-level debugger, and cross-assemblers and cross-compilers for the SDS 8051 are described on the next page. Each package supports most 8051-derived microcontrollers including the 80C51, 83C053, 83C054, 80C451, 80C528, 80C550, 80C552, 80C562, 80C592, 80C652, 80C654, 83C751, 83C752, 80C851, 80C852, and 8XCL410/710.

TYPICAL EMULATION PROBE



Stand-alone debug station for 80C51/8051-based systems

SDS 8051

SYMBOLIC DEBUGGING PACKAGE (XRAY51)

The XRAY51 debugger helps to locate programming errors in the source code of C,PL/M or assembly language programs for the 8051 family. With XRAY51, the user can isolate these errors by controlling and monitoring program execution using the same high-level or assembly level terms, definitions and structures found in the original source program. For example, the user can single-step through the program a specified number of microcontroller instructions or high-level language lines. Variables can be accessed with respect to the source language in which they had been defined. Operating XRAY51 in Assembly mode, the user can manipulate the contents of all processor registers. Only those registers that are part of the selected 8051 derivative are allowed to be accessed.

Macros may be defined that can execute complex user command procedures and provide a variety of complex breakpoints.

When debugging with XRAY51, the user can examine the contents and modify the value of any variable, compute the value of C,PL/M source language expressions and assembly level address expressions, and define, remove, or display symbols.

Command files can be used to direct XRAY51 to read or write simulated microprocessor input/output from or to a file, allowing easy implementation of automated test sequences. Command files enable scripts of debugger commands to be processed automatically without the need of user interaction.

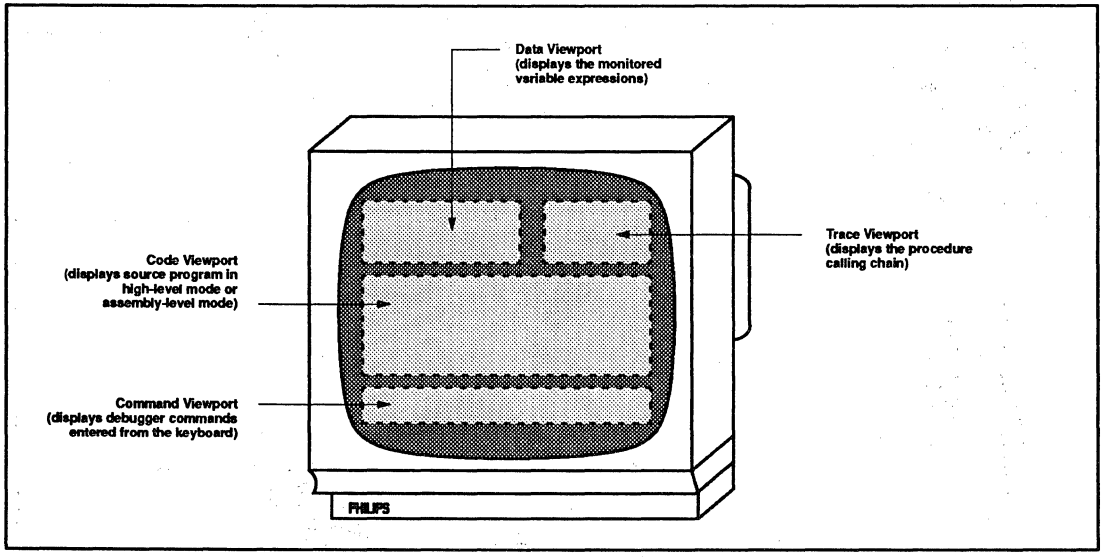
XRAY51 also allows a virtually unlimited number of user-defined windows.

A simulator version of XRAY51 having the same features as the emulator is available.

Features

- Integrated C,PL/M-51 source-language and assembly-language debugging; toggle by function key at any time
- Window-oriented display with well-organized segregation of debugging information
- Symbolic debugging with C,PL/M-51 variables and C,PLM-51 statements
- Simple and complex breakpoints
- Single-step execution
- User-definable screens and viewports—ability to write selected information
- Command macros
- Breakpoint macros (limited to the number of breakpoints of the SDS 8051)
- Command and breakpoint macros may contain C statements, including: FOR, WHILE DO, PRINTF, and FPRINTF
- On-line context-sensitive help
- Command files
- Output logging
- Fully supports SDS 8051 with the latest firmware version
- High-level trace
- Transparent ICE mode for direct control of SDS 8051

HIGH-LEVEL XRAY51 SCREEN



Stand-alone debug station for 80C51/8051-based systems

SDS 8051

CROSS-ASSEMBLERS

The ASM51 cross-assembler* is available for translating 8051 assembly language programs into relocatable object code:

- It accepts Intel-compatible assembler source programs and produces relocatable '–.OBJ' object files. An absolute or executable 'A.OUT' load-image is obtained using the LINK51 linker.

The assembler is a powerful development tool with many directives, so a source file is easily composed, and translation to code is fast.

The assembler directives support features such as the inclusion of ASCII strings, checking module length, forcing a start address, allocating bytes to labels used in current or other modules and module definition.

A very powerful feature is the macroprocessor which includes tools such as macro-substitution, file inclusion and conditional assembly.

ASM51 Features

- Produce relocatable object code, listing files and diagnostic messages
- Accept Intel-compatible source programs
- Compatible with Tasking's PL/M-51 compilers
- Conversion to IEEE object code format
- Compatible with XRAY51 High-Level/Assembly-Level Debugger
- Include many utilities such as, Librarian, Cross-reference generator, object code-convertors
- Support segment overlay at the Assembly level
- C and MPL (Macro Programming Language) compatible macro preprocessors included
- Separate linking phase

CROSS-COMPILERS

Two cross-compilers* are available for efficiently translating programs written in the PL/M-51 language into 8051 Assembly:

- **PLMT151:** generates Intel-compatible source, which can be assembled and linked using the ASM51 and LINK51 programs.
- **C-51:** generates Intel-compatible assembly source, which can be assembled and linked using ASM51 and LINK51.

Both compilers are always supplied together, allowing you to choose the one that suits your requirements best.

PLMT151 Features

- Fast, single-pass, memory-based compilers that are fully compatible with the Intel PL/M-51 language definition
- Fully compatible with ASM51/LINK51
- Produce highly optimized code
- Support the XRAY51 high-level debugger
- Support in-line assembly code
- Optional IEEE single-precision floating point package
- Include several utility programs: pr, grep, aar, cpp, scf_i51, ocf_o51, oct_ihex, oct_ieee, oct_srec, ocf_ihex, ocf_srec, makelib and pack
- Utility library included in source code
- Combiner performs optimization at load-file level
- Easy tailoring to application and target hardware
- Produce relocatable code and data
- Support optimize(4): intermodule overlay optimization
- Support 15 interrupts
- Automatic installation with self-test
- Available for many hosts besides the IBM PC (MS-DOS), for example: (micro-)VAX, (VMS, Ultrix), HP9000/300 (HP-UX), SUN-3 (Sun-OS); available from BSO/Tasking Software B.V.

C-51 Features

- Supports ANSI C with powerful 8051 extensions:
  - *bit*-type to use the on-chip bit addressable area
  - *interrupt <number>* and *using <registerbank>* for interrupt servicing at C-level
- Generates Intel-compatible Assembly source
- Supports HLL-debugger XRAY51 (in combination with ASM51)
- Supports all members of the 8051 family
- Produces very efficient and reliable code
- Code is ROMable
- Easy migration from BSO/Tasking PL/M-51 to BSO/Tasking C-51 through mixed language programming
- Supports 4 memory models to best meet application requirements
- Separate compilation of program modules
- Embodies integral standard preprocessor
- Is one-pass, fast and compact (no intermediate code or files)
- 3-layer design simplifies maintenance
- Complete set of UNIX-like compiler options
- Comes with C library and run-time support, including I/O calls (+ *printf*), memory management, floating point math, and arithmetic functions
- Available on many hosts: (micro)VAX (VMS or Ultrix), IBM PC (DOS, Xenix), IBM 6150, SUN-3/-4/-386i (SUN-OS), HP9000-300 (HP-UX), Apollo (UNIX)
- Generates re-entrant + relocatable code and relocatable data
- Same calling sequence on all hosts
- Same source and generated assembly code can be used on all hosts
- Automatic installation and self test
- Benchmarks and application notes are available on request.

* Sourced from BSO/Tasking Software B.V. Amersfoort, The Netherlands

Stand-alone debug station for 80C51/8051-based systems

XRAY51 AND SDS COMMANDS

XRAY51 Commands

XRAY51 uses a powerful command language that employs C language expressions. Note, all commands can be issued in an abbreviated form.

Session Control Commands

HOST	Enter the host operating system environment
LOAD	Load an object module for debugging
QUIT	Terminate a debugging session

Execution and Breakpoint Commands

BREAK INSTRUCTION	Set an instruction breakpoint
CLEAR	Clear a breakpoint
GO	Start or continue program execution
GOSTEP	Execute macro after each instruction step
STEP	Execute a specified number of instruction lines
STEPOVER	Step, but execute through procedures

Display Commands

DISASSEMBLE	Display disassembled memory (assembly mode)
DUMP	Display memory contents
EXPAND	Display all local variables of a procedure
FIND	Search for a string
FOPEN	Open a file or device for writing
FPRINTF	Print formatted output to a viewport or file
LIST	Display source code
MONITOR	Monitor variables
NEXT	Find next occurrence of a string
NOMONITOR	Discontinue monitoring variables
PRINTF	Print formatted output to a command viewport
PRINTVALUE	Print the value of a variable

Memory Commands

COMPARE	Compare two blocks of memory
COPY	Copy a memory block
FILL	Fill a memory block with values
SEARCH	Search a memory block for a value
SETMEM	Change the values of memory locations
SETREG	Change the contents of a register
TEST	Examine memory area for invalid values

Port I/O and Interrupt Commands

DIN	Display input port buffer values
DOUT	Display output port buffer values
INPORT	Set or alter input port status
INTERRUPT	Simulate an interrupt
NOINTERRUPT	Cancel pending interrupts
OUTPORT	Set or alter output port status
RIN	Rewind input file associated with input port
ROUT	Rewind output file associated with output port

Symbol Commands

ADD	Create a symbol
DELETE	Delete a symbol from the symbol table
PRINTSYMBOLS	Display symbol, type, and address
SCOPE	Specify current module and procedure scope

Utility Commands

CEXPRESSION	Calculate the value of an expression
ERROR	Set include-file error handling
HELP	Display on-line help screen
INCLUDE	Read in and process a command file
JOURNAL	Record a debugger session in a file
LOG	Record debugger commands and errors in a file
MODE	Select debugger mode (high or assembly)
OPTION	Set debugger options for this session
PAUSE	Pause simulation
RESET	Simulate microprocessor reset
RESTART	Restart program counter to the program starting address
STARTUP	Save the default startup options

Macro Commands

DEFINE	Create a macro
SHOW	Display the macro source

Viewport Commands

VACTIVE	Activate a viewport
VCLEAR	Clear data from a viewport
VCLOSE	Remove a user-defined viewport or screen
VMACRO	Attach a macro to a viewport
VOPEN	Create a screen or viewport or change sizes
VSCREEN	Activate a screen
VSETC	Set the cursor position for a viewport
ZOOM	Increase or decrease the size of a viewport

Function Key Commands

VACTIVE–1	Activate the next viewport (counter clockwise)
VACTIVE+1	Activate the next viewport (clockwise)
MODE	Change debugging mode (assembly/high)
ZOOM	Increase or decrease the size of a viewport
HELP	Access on-line help
VSCREEN	Change the active screen
BACK UP	Back up one command
STEP	Execute one machine instruction or source line
STEP OVER	Step, but execute through procedures

In-Circuit Emulator Commands

ICE	Communicate with in-circuit emulator (ICE)
NOICE	Return to debugger command mode

SDS Commands

Power-Up Commands

RESET	Back to initialization

Program Execution Commands

BRn	Break at given address (n = 0, 1, 2, 3)
BRR	Break within given address range
BRB	Break at branch instruction
BV	Break on internal RAM value
GO (FROM, TILL)	Initiates program execution (specified addresses)
STEP (FROM)	Executes single instruction or instructions (from a specified address)
UD	User-defined memory address on display
TRACE	Display executed instruction flow (real time)
INT	Display interrupt enable, priority and status of all interrupt sources

Memory Access Commands

DBYTE	Displays specified byte from internal data memory
XBYTE	Displays specified byte from external data memory
CBYTE	Displays specified byte from code memory
RBYTE	Displays specified byte from on-chip register memory
RBIT	Displays specified bit from on-chip bit-addressable memory
ASM	Assemble single instruction mnemonic into program memory
DASM	Disassemble memory values into mnemonics

Memory Set Commands

All registers can be set and displayed by commands equal to their names (PSW, SCON, P3, TH1, TL1, TL0, etc.)

Serial I/O Interface Commands

SAVE	Copies data from SDS 8051 program memory to host disk file
LOAD	Transfers file from host to SDS 8051

Stand-alone debug station for 80C51/8051-based systems

SDS 8051

SDS 8051 SPECIFICATION

RS232C	– Two ports – Baud rate selectable from 300 to 19200 baud – Download file format: Intel HEX – Recognize Xon/Xoff
Trace memory	– 2048 lines deep, 64 bits wide – Internal/external code memory fetches; data from all ports, lables, user test clips – Selective tracing on cycle type
Emulation memory	– 64 kbytes – No wait states
Clock speed	– Up to 20MHz, real time
Power down	– Supports power-down and idle mode
Signals to external equipment	– Output from SDS: ALE: indicates valid address CLK: indicates opcode read PSENE: indicates byte read EMUL: indicates running user program – Input to SDS: EXTBRK: stop emulation by external pulse
Size	– 300 × 66 × 235 mm (W × H × D)
Weight	– 5 kg (approx.)
Power supply	– 110/220V AC, 50/60 Hz
Cables	– Mains cable; RS232/V24 cable for connection to an IBM PC/XT

ORDERING INFORMATION[1]

All of the 8051 development tools listed below are available from your local Philips Components sales office. The symbolic debugging package, cross-assembler and compiler are sourced from Tasking Software B.V., Amersfoort, The Netherlands, which holds all intellectual property rights for these products.

Stand-alone debug station (excluding probe) for the 8051 family	OM4120S
Symbolic debugging package XRAY51	OM4129
Cross-assembler for MS-DOS (comprises the as8051 and the Intel-compatible ASM51)	OM4142
PL/M-51 compiler for MS-DOS (comprises the plm51 and the Intel-compatible PLMT151)	OM4144
C-51 compiler	OM4136
Emulation probes[4]:	
for 8032/8052/80C32/80C52[3]: (DIL/PLCC interface)	OM4111+ OM4110
for 80C51/80C31/8051/8031: (DIL/PLCC interface)	OM1092+ OM1097
for 80CL51:	OM1079
for 8XC053/054: (SDIP interface)	OM5054
for 8XCL410/710: (DIL interface)	OM1079
for 8XC451: (PLCC interface)	OM4123
for 8XC528: (DIL/PLCC interface)	OM4111+ OM4110
for 8XC550: (DIL/PLCC interface)	OM5055+ OM4110
for 8XC552/562: (PLCC interface)	OM1092+ OM1095
for 8XC592: (PLCC interface)	OM4112+ OM4110
for 8XC652/654: (DIL/PLCC interface)	OM1092+ OM1096
for 8XC751: (DIL/PLCC interface)	OM1094
for 8XC752: (DIL interface)	OM5072
for 8XC851: (DIL/PLCC interface)	OM1092
for 83C852:	OM4119
Adapters:[2]	
for 8XC451: 68-pin PLCC probe to 64-pin DIL socket	OM4124
for 80C51: 40-pin DIL probe to 44-pin PLCC socket	OM4125
for 83C852: contactless card reader adapter	OM4118/1
ISO contact card reader adapter	OM4118/2
AFNOR contact and reader adapter	OM4118/3
Conversion kits: for converting the OM1092 to an:	
8XC552/562 probe	OM1095
8XC652/654 probe	OM1096
80C31/80C51 probe	OM1097

NOTES:

1. A minimum SDS configuration must include an OM4120S and an emulation probe. A minimum PC configuration must include 256 kbytes system memory running MS-DOS 3.0 (or later releases), one floppy disk drive and a monochrome monitor. However, we recommend using an IBM PC/XT (or compatible) with 640 kbytes RAM, hard disk drive, floppy disk drive and a color monitor. The SDS software can be supplied on $3^1/_2$" or $5^1/_4$" diskettes. Conversion kits require the OM1092.
2. Support for QFPs will be available in the near future.
3. Minor restrictions.
4. All probes 16MHz, except OM4123 (12MHz).

™ ICE is a trademark of Intel Corporation.
XRAY is a trademark of Microtec Research Inc.
MS-DOS is a trademark of Microsoft Corporation.
VAX is a trademark of Digital Equipment Corporation.

Cross-assembler package for 80C51/8051-based systems

OM4142 (ASM51)

This package comprises the cross-assembler for translating 8051 assembly language programs into relocatable object code:

- The ASM51* macroassembler which accepts Intel-compatible assembler source programs and produces relocatable '–.OBJ' object files. An absolute or executable 'A.OUT' load-image is obtained using the LINK51 linker.

The absolute object file can be modified to IEEE format to serve as input to the XRAY51 debugger.

FEATURES

- Produces relocatable object code, listing files and diagnostic messages
- Accepts Intel-compatible source programs (ASM51)
- Supports most 8051-derived microcontrollers including the 80C51/31, 87C51, 8XC451/528/552/562/592/652/654/751/851, 8XCL410/710 and the 8051/31/52/32.
- Compatible with PL/M-51 compilers
- Conversion to IEEE object code format
- Compatible with XRAY51 High-Level/Assembly-Level Debugger
- Includes many utilities such as, Librarian, Cross-reference generator, object code-convertors
- Supports segment overlay at the Assembly level
- C and MPL (Macro Programming Language) compatible macro preprocessors included
- Separate linking phase.

OPERATION

The input to both assemblers usually comes from a preprocessor which interprets preprocessor directives in the source program to deal with file inclusion, macro definition and replacement, conditional text inclusion, etc. Two preprocessors are available as separate programs, allowing the programmer to use either the C preprocessor directives or Intel's Macro Programming Language, or even a mixture of both.

The output may then be assembled using the ASM51. Depending on the selected member of the 8051 family, the ASM51 assembler enables or disables the names of special function registers that are applicable. For the ASM51, this is easily solved by target-dependent inclusion of a specific equate list in the source file.

The assembler translates a source program into relocatable object code using three different passes. The program syntax, assembler directives and user-defined symbols are checked and processed during the first pass. In the second pass, all generic forward jumps and calls are optimized. In the third pass, relocatable code is generated.

To obtain a single executable load image, all necessary relocatable objects, including library modules, are linked together using the proper linker (i.e., LINK51). This executable load image may then be converted to an ASCII file that may be downloaded into an EPROM programmer or an emulator.

The assembler is capable of generating symbolic debug information to accommodate information for the XRAY debugger.

ASSEMBLER LIMITATIONS

The number of user-defined symbols and macro parameters is limited only by the available heap space. Save/restore nesting is restricted to 16 levels.

DIFFERENCES BETWEEN PHILIPS' ASM51* AND INTEL'S ASM51

Unlike Intel's ASM51, which restricts the use of register equates to the A and R0–R7 registers, Philips' ASM51 puts no constraints on register name assignment. And it can optimize generic JMP and CALL instructions, even when they contain a forward reference. In such cases, Intel's ASM51 will always produce code for a LJMP and LCALL, which takes 50% more code.

Philips' ASM51 supports four new directives that are not available in the Intel ASM51:

- $listall — generate a listfile in every pass (not only in the final one). This improves diagnostics.
- $(no-)optimize — enable (default) or disable generic JMP/CALL optimization.
- $debuginfo — control symbolic debug information generation.

Since the $gen, $genonly, $nogen, $include and $macro directives are already dealt with during the preprocessing stage, these can be ignored by Philips' ASM51.

Similarly, $(no-)xref and $(no-)symbols directives are ignored—report utilities are available to accomplish the same task. And the $workfiles is no longer useful and is ignored too.

Philips' ASM51 recognizes ?SYMB, ?LINE and ?FILE symbols that are used to pass debug information towards the object module. Philips' ASM51 recognizes the C-like #line directive to adjust the line number and file name. This directive is generated by both preprocessors to synchronize the output line with the original input. This improves the error diagnostics of Philips' ASM51 compared with those of its competitors, since error messages now refer to the proper (include) file and line number.

A powerful addition is the overlay() which gives the programmer full control of the section overlay strategy.

UTILITY PACKAGE

Several utility programs are included for your convenience. Amongst these are a C preprocessor (cpp) and an Intel MPL-compatible preprocessor (mpl) to deal with the various preprocessor directives and macros.

In many cases, the format of the object code produced is not suitable for EPROM-programmers, emulators or debuggers. Therefore, several utilities are included to convert the code to Intel HEX (oct_ihex), Motorola S0–S9 (oct_srec) or IEEE-695 (oct_ieee) or vice-versa (ocf_ihex, ocf_srec).

An optional utility package is available separately from Tasking Software B.V. It contains UNIX-like utilities to obtain a (cross-reference) list of all user-defined symbols in a program ('axref' and 'anm'), and the 'asize' utility to get information about the section sizes.

HARDWARE/SOFTWARE REQUIREMENTS AND INSTALLATION

OM4142 software comes to you on $5^1/_4$" diskettes ($3^1/_2$" diskettes are available on request) together with extensive documentation, including two Assembler Reference Guides, Preprocessor Manuals, Utility and Installation Guide. The software requires an MS-DOS computer with a hard disk and at least 512k byte RAM installed. Software installation is simple, well-documented and can be verified afterwards using an automatic verification program.

* Sourced from BSO/Tasking Software B.V., Amersfoort, The Netherlands, which holds all intellectual property rights for this software.

Cross-assembler package for 80C51/8051-based systems

OM4142 (ASM51)

RELATED PRODUCTS

The OM4142 assembler is part of a complete programming and development package for the 8051 family of microcontrollers. For both assemblers, there is an excellent PL/M-51 compiler available from Philips offering the convenience and benefits of high-level language programming, resulting in a dramatic increase of programmer productivity. Testing and debugging can be accelerated using the High-Level/Assembly-Level XRAY51 debugger. Contact your local Philips Semiconductors software sales office for more information. Ask for the following leaflets:

Symbolic debugging package XRAY51 for the SDS 8051 emulator, ordering code 9398 366 10011;

PL/M-51 compiler package for 80C51/8051-based systems, ordering code 9398 366 20011;

Stand-alone debug station for 80C51/8051-based systems, ordering code 9398 366 00011.

SOFTWARE MAINTENANCE AND SUPPORT

After a 90-day warranty period, in which all support will be given free of charge, a software update and support agreement can be taken out with Philips for a modest annual fee. Philips realizes your need for fast and comprehensive support, and with a support license, you get direct access to a development team that can solve your problems.

ORDERING INFORMATION

TYPE NUMBER	DESCRIPTION
OM4142	8051 cross-assembler package comprising the ASM51 assembler, and the LINK51 linker
Related Products	
OM4144	PL/M-51 compiler package comprising the compiler
OM4129	XRAY51 high-level debugger for the SDS 8051

Orders can be placed via your local Philips Semiconductors sales representative.

™ Intel is a trademark of Intel Corporation.
XRAY is a trademark of Microtec Research Inc.
MS-DOS is a trademark of Microsoft Corporation.
UNIX is a trademark of AT&T Bell Laboratories..

PL/M-51 compiler package for 80C51/8051-based systems

OM4144 (PLMTI51)

This package comprises the cross-compiler (PLMTI51*) which efficiently translates programs written in the PL/M-51 language into 8051 Assembly. The PLMTI51 generates Intel-compatible source, which can be assembled and linked using Philips' ASM51 and LINK51 programs.

This compiler embodies the latest techniques. It is built upon a YACC-based parser. Intermediate code is built using tree structures that are optimized into new trees by the compiler before code-generation starts. An internal peephole optimizer further improves the density of the generated code.

It can be ordered under the Philips' type number OM4144.

FEATURES

- State-of-the-art compiler: fast, single-pass, memory-based
- Fully compatible with the Intel PL/M-51 language definition
- Fully compatible with Philips' ASM51/LINK51 cross-assembler and linker
- Produces highly optimized code
- Supports most 8051-derived microcontrollers, including the 80C51/31, 87C51, 8XC451/528/552/562/592/652/654/751/851, 8XCL410/710/51 and the 8051/31/52/32
- Both compilers support the XRAY51 high-level language debugger, available from Philips
- Supports in-line assembly code
- Optional IEEE single-precision floating point package
- Includes several utility programs: pr, grep, aar, cpp, scf_i51, ocf_o51, oct_ihex, oct_ieee, oct_srec, ocf_ihex, ocf_srec, makelib and pack
- Utility library included in source code
- Combiner performs optimization at load-file level (Philips' ASM51 only)
- Easy tailoring to application and target hardware
- Produces relocatable code and data
- Available for many hosts besides the IBM PC (MS-DOS), for example: (micro-)VAX, (VMS, Ultrix), HP9000/300 (HP-UX), SUN-3 (Sun-OS); available from BSO/Tasking Software B.V.
- Supports optimize(4): intermodule overlay optimization
- Supports 15 interrupts
- Automatic installation with self-test
- Benchmarks available on request

PL/M-51

PL/M-51 is a structured, high-level programming language derived from the Intel PL/M-80 language and adapted to the specific capabilities of the 8051 family of single-chip microcontrollers. It supports Boolean processing and allows efficient access to all microcontroller hardware functions.

Software development in PL/M-51 combines the ease of programming in a high-level language with access to all of the 8051 I/O and memory—features that are normally only available to assembly language programmers.

The Philips' implementation of PL/M-51 is an extremely fast single-pass optimizing compiler that is fully compatible with the Intel PL/M-51 language definition.

IMPLEMENTATION-DEPENDENT DATA

Data Types

- Data types BIT, BYTE and WORD are allowed for variables, arrays, structures or combinations thereof.
- Memory types MAIN, IDATA, REGISTER, AUXILIARY and CONSTANT are supported.
- Data can be stored at fixed locations using AT, or stored dynamically using BASED pointer variables.

Procedures

- BIT, BYTE or WORD typed procedures, returning a value upon completion
- Untyped procedures, invoked with a CALL statement
- Optional procedure attributes:
  - USING(n), specifies the register bank (n) to be used by the procedure.
  - INTERRUPT(n), defines an interrupt procedure for interrupt (n).

Statements

- Control statements: do while ... end, do ... to ... by ... end
- Conditional statements: if ... then ... else ...
- Miscellaneous statements: do ... end, do case ... end, call, goto, enable/disable

Library Routines

Internal (Built-in) Procedures:

LENGTH	PROPAGATE
WORD	ROL
EXPAND	SCR
SHR	SIZE
SCL	BOOLEAN
TIME	SHL
LAST	ROR
DOUBLE	TESTCLEAR

PLM51.LIB:
Library routines used by the compiler.

UTIL51.LIB or util51.oa, util51:
Assembler utility libraries for both compiler versions, consisting of procedures for string manipulation. It contains the routines MOV, RMV, CMP, FNDB, FNDW, SKPB, SKPW, SETB and SETW for each memory type and for each register bank.

Re-entrancy

The generated code is not re-entrant, because PL/M-51 is not defined as a language with re-entrant procedures. Because of the limited size of internal RAM, the local and formal parameters of procedures are not put on the stack, but on static areas, which can be overlayed by the compiler using $OPTIMIZE(3).

Compiler Controls

ROM(s), REGISTERBANK(n), OPTIMIZE(n), (NO)INTVECTOR, (END)ASM, INCLUDE, SAVE/RESTORE, (NO)LIST, (NO)SOURCE, (NO)CODE, (NO)DEBUG, (NO)OBJECT, SET/RESET, EJECT and IF/ELSIF/ELSE/ENDIF

Differences Between Philips' and Intel's PL/M-51

- Philips' PL/M-51 allows bit-structures to be ATed at byte-variables in bit-addressable memory
- Philips' PL/M-51 supports floating point
- Invocation and compiler controls are mostly different
- Object modules are not generated
- Error messages are mostly different
- Different assembly language interfacing (only for Philips' plm51)
- Compiler limits always the same or better than Intel's

* Sourced from BSO/Tasking Software B.V., Amersfoort, The Netherlands, which holds all intellectual property rights for this software.

PL/M-51 compiler package for 80C51/8051-based systems

OM4144 (PLMTI51)

Code Optimization

Five levels of optimization are supported:

- Level 0: performs folding of constant expressions and of address calculations, in case a constant offset exists.
- Level 1: performs all of level 0 plus elimination of unreachable code, strength reduction of expressions and partial condition evaluation at runtime.
- Level 2: performs all of levels 0 and 1 plus machine code (peephole) optimizations and register history.
- Level 3: performs all of levels 0, 1, and 2 plus automatic overlaying of on-chip RAM variables.
- Level 4: performs all of the other levels plus support of intermodule overlay of on-chip data.

General Optimizations

- Store-copy optimization
- Local constant propagation
- Register allocation
- Peephole optimization
- Dead code elimination
- Constant folding
- Index simplification

Object-code Optimizations

- Branch optimization (sjmp, ajmp, ljmp, acall, lcall)
- Effective address optimization

8051-Specific Optimizations

- Optimal use of the range of address modes of the 8051 family
- Overlay of local data and formal parameters done by $OPTIMIZE(3)/(4)

Easy Adaptation to Target Environment

Cross-compilers are used to develop embedded microprocessor applications, where the hardware environment in which they will run is not fixed beforehand. Adaptation to the target hardware environment takes place via the files headx. These files are used to define all system-dependent set-ups such as the power-on-restart vector, the initial stackpointer value, the definitions of segments needed by the PL/M-51 code and the mapping of those segments to the physical addresses. The files can be modified by the user to match his specific needs.

The PL/M-51 cross-compiler can accommodate different target environments, because:

- The location of the stack is held in a special target set-up file (head_xx file) which can be changed to match your requirements
- This target set-up file also allows general housekeeping tasks to be executed before program start-up
- Simple interfacing to target operating systems and low-level I/O and system routines
- Code and data segments can be placed anywhere in the 64kbyte data and address space of the 8051 processor
- Efficient calls to library routines can be in PL/M or assembler (in-line)
- A user-written interrupt handler can be made by specifying $NOINTVECTOR
- The generated code is ROMable.

Restrictions

The PL/M-51 compiler has a few restrictions listed here for completeness:

- Nesting of all LITERALLY invocations: 8
- Nesting of INCLUDE controls: 8
- Nesting of blocks: 32
- Number of elements in a factored list: no limit
- Number of characters in an input line: 160
- Number of switch names (conditional compilation): 20
- Length of a string constant: 254
- Number of cases in a DO CASE block: 84
- Number of EXTERNAL items: no limit
- Number of non-EXTERNAL procedures in module: no limit
- Number of names in a module: memory dependent

UTILITY PACKAGE

Several utility programs are included for your convenience. Amongst these are a C preprocessor (cpp), an Intel MPL compatible preprocessor (mpl)—both part of the 8051 Cross-Assembler—to deal with the various preprocessor directives and macros. Pagination and pattern search commands 'pr' and 'grep' come with the PL/M-51 compiler package.

In many cases, the format of the produced object code is not suitable for EPROM-programmers, emulators or debuggers. Therefore, several utilities are included (in the 8051 cross-assembler) to convert the code to Intel HEX (oct_ihex), Motorola S0–S9 (oct_srec) or IEEE-695 (oct_ieee) or vice-versa (ocf_ihex, ocf-srec) or to archive object modules (aar).

OPTIONAL SOFTWARE

An optional utility package is available separately from BSO/Tasking Software B.V. It contains UNIX-like utilities to obtain a (cross-reference) list of all user-defined symbols in a program ('axref' and 'anm'). Part of this package is the 'asize' utility to get information about the section sizes, 'asort' to sort or merge files, 'astrip' to remove symbols and relocation information, 'adump' to display the contents of an object file.

Also available is an IEEE floating-point library for the 8051 family.

HARDWARE/SOFTWARE REQUIREMENTS AND INSTALLATION

OM4144 software comes to you on $5^1/_4$" diskettes ($3^1/_2$" diskettes are available on request) together with the extensive *PL/M-51 Application Manual*.

The software requires an MS-DOS (Rel. 3.0 or higher) computer with a hard disk and at least 512kbyte RAM installed. Software installation is simple, well-documented and can be verified afterwards using an automatic verification program.

PL/M-51 compiler package for 80C51/8051-based systems

OM4144 (PLMTI51)

RELATED PRODUCTS

The OM4144 compilers are part of a complete programming and development package for the 8051 family of microcontrollers, a package which includes a cross-assembler and (optimizing) linker. Testing and debugging can be accelerated using the high-level/assembly-level XRAY-51 debugger in combination with Philips' SDS 8051 emulator, or the 8051 Simulator (available from Tasking software B.V.). Contact your local Philips Semiconductors sales office for more information.

Ask for the following leaflets:

Symbolic debugging package XRAY51 for the SDS 8051 emulator, ordering code 9398 366 10011;

Cross-assembler package for 80C51/8051-based systems, ordering code 9398 366 30011;

Stand-alone debug station for 80C51/8051-based systems, ordering code 9398 366 00011.

More information about the PL/M-51 Language can be found in the *PL/M-51 User's Guide*, available from Intel (ordering code: 121966-003).

SOFTWARE MAINTENANCE AND SUPPORT

After a 90-day warranty period, in which all support will be given free of charge, a software update and support agreement can be taken out with Philips for a modest annual fee. Philips realizes your need for fast and comprehensive support, and with a support license, you get direct access to a development team that can solve your problems.

ORDERING INFORMATION

TYPE NUMBER	DESCRIPTION
OM4144	PL/M-51 compiler package comprising the PLMTI1 compiler
Related Products	
OM4142	8051 cross-assembler package comprising the ASM51 assembler, and LINK51 linker
OM4129	XRAY51 high-level language debugger for the SDS 8051

Orders can be placed via your local Philips Semiconductors sales representative.

™ Intel is a trademark of Intel Corporation.
MS-DOS and XENIX are trademarks of Microsoft Corporation.
XRAY is a trademark of Microtec Research Inc.
UNIX is a trademark of AT&T Bell Laboratories.
VAX, microVAX, VMS and Ultrix are trademarks of Digital Equipment Corp.
HP9000 and HP-UX are trademarks of Hewlett Packard Co.
SUN-3 and SunOS are trademarks of Sun Microsystems Inc.
IBM PC is a trademark of International Business Machines Corp.

8051 C cross-compiler

OM4136

INTRODUCTION

The BSO/Tasking C-51 cross-compiler (OM4136) offers a new approach to high-level language programming for the 8051 family. It is a very powerful combination of our extensive 8051 knowledge and our ANSI compliant C compilers. the result of this combination is a compiler that is fast, efficient and supports all the members of the 8051 family.

FEATURES

- Supports:
  - Full ANSI Standard
  - All members of the 8051 family
  - Reentrant programming
  - Extremely efficient and powerful pointer arithmetic
  - Inline assembly programming
  - Inline expansion of predefined functions (_rol, _ror, etc.)
  - Full Intel OMF51 and IEEE695 version 4.0 object formats
  - Data overlay mechanism
  - Full calling interface (with parameter passing) to BSO/Tasking PL/M-51 (OM4144)
  - Fast 8-bit *char* arithmetics
  - C-level access to chips' SFRs
  - C-level bit-type and interrupt
- Generates Intel-compatible assembly source
- Outputs symbolic information for source level debugging using XRAY51
- Produces very efficient, fast executing and reliable code
- Code is ROMable
- Supports 4 memory models to best meet application requirements
- Separate compilation of program modules
- Embodies integral standard preprocessor
- Is one-pass, fast and compact (no intermediate code or files)
- 3-layer design simplifies maintenance
- Complete set of UNIX-like compiler options
- Comes with C library and run-time support, including I/O calls (+ *printf*), memory management, floating point, math and arithmetic functions
- C libraries in source code included for learning and tuning purposes
- The OM4136 product is for IBM PC DOS as host. From BSO/Tasking, the C-compiler is also available on the hosts: (micro)VAX (VMS or Ultrix), IBM PC (Xenix), IBM PS/2 (OS/2, AIX), IBM RISC System/6000, SUN-3/-4 (SUN-OS), HP9000/300 (HP-UX) and Apollo (UNIX)
- Generates reentrant and relocatable code and relocatable data
- Same calling sequence on all hosts
- Same source and generated assembly code can be used on all hosts
- Automatic installation and self test
- Benchmarks and application notes are available on request

C LANGUAGE

Although intended as a general purpose high-level programming language, C is perhaps most powerful in the area of real-time system programming for embedded microcontrollers.

ANSI C features economy of expression, is well defined, has an improved type checking, supports structured and modular programming and has a rich set of data- and operator-types that map to virtually every single address space microcontroller/processor. Since the operators and type definitions generally match well with the instruction sets and word lengths of most microcontrollers, C is very efficient in therms of code size and execution speed.

COMPILER TECHNOLOGY

The 8051 C compiler was built using the latest compiler technology, including optimization and multi-layering. It is essentially a one pass compiler, translating on a function-by-function basis, achieving very fast compilation and a large span for the optimization process. A significant effort was made to obtain a well-defined, layered product.

The top layer basically consists of a lexical and grammar analyzer with co-routine structures to serve both the pre-processing and compilation needs. The grammar analyzer is produced by an LALR(1) parser generator, eliminating a potential source of error.

A second layer separates the actual code generation layer from the top layer and is merely intended to simplify re-targeting of the compiler to other target processors. It was designed to represent the translated program independent from the language and the target. This is where most of the optimizations are done.

The code-generation layer is driven both by the second layer and by a set of tables and rules reflecting the target assembly language and the behavior of individual instructions.

IMPLEMENTATION DEPENDENT DATA

This section outlines features of the C-51 compiler which, for the most part, are specific to the 8051 family of microcontrollers.

Compiler Options

The following options are supported:

-a specify function parameter size
-b specify default register bank number
-C specify 8051 derivative type
-D define named identifier to the preprocessor
-E only run the preprocessor
-f read command line options from file
-g generate high-level debugging information for XRAY51 hll-debugger
-I force preprocessor to look for include files in the specified directory
-m specify memory sizes for static allocation checking
-M select memory model
-n write output to screen instead of output file
-o write output to named output file
-O controlled optimization
-r specify ROM size for optimal jump-type generation
-R control segment assignment
-s merge C source with generated assembly code
-S put strings in ROM only
-t do not produce module summary information
-U remove any initial definition of named identifier
-v do not generate code for interrupt vectors
-w suppress warning messages

Data Sizes

Type	Size (in bytes)
bit	1 bit
char	1
short int	2
int	2
long int	4
float, (long) double	4 (IEEE single precision)
pointer type	1 or 2*

*NOTE: Pointers to *data*, *idat* and *pdat* have a size of 1, whereas pointers to *rom*, *xdat* and functions have a size of 2.

8051 C cross-compiler

OM4136

Code Optimization

The compiler intermediate layer performs general code-optimization and the code-generator does some final optimizations that are specific to the 8051.

General Optimizations:

- Register allocation
- Branch optimization
- Dead code elimination
- Constant folding
- Arithmetic simplification

Code Generator Optimizations:

- Store-copy optimization
- Peephole optimization

Adaptation to Target Environment

Cross-compilers are used to develop embedded microcontroller applications, where the hardware environment in which they will run is not fixed in advance. The 8051 compiler allows flexible adaptation to different target environments:

- The compiler supports 4 different memory models. This allows the compiler to generate the best possible code for a particular hardware implementation.
- Interrupt service routines can be written in C or assembly (through special controls)
- Generated code is ROMable
- 8051 derivative is switch selectable
- User-controlled mapping of code and data

Pragmas

The *#pragma* directive is intended to supply target dependant data to the compiler, without violating the C language. In C-51, pragmas are used to merge C lines with generated assembly, to control code generation for interrupt service routines and to support inline assembly programming.

LIMITATIONS

The use of several C constructs is restricted to a maximum number, as proposed by ANSI. Both the preprocessor and compiler meet these 'environmental requirements of a conforming implementation'. As an additional constraint however, the number of statements in a function body is restricted to approximately 1800.

DOCUMENTATION

The compiler is shipped with the 8051 C Compiler User Manual.

Additional information about the C language can be found in "The C Programming Language" (second edition), by B. Kerighan and D Ritchie (1988, Prentice Hall).

RELATED PRODUCTS

The C51 compiler (OM4136) is part of a complete programming and development package for the 8051 family of microcontrollers. This incorporates a HLL debugger, XRAY-51 (OM4129) in combination with Philips' SDS-8051 emulator, PL/M-51 compiler (OM4144), cross-assembler and (optimizing) linker (OM4142), all available from Philips Semiconductors.

SOFTWARE MAINTENANCE AND SUPPORT

After a 90 days warranty period, in which all support will be given free of charge, a software support contract is availabe from BSO/Tasking at a modest annual fee. BSO/Tasking realizes the need for fast and thorough support in this complex area where a lot of time may be wasted before a (afterwards simple) problem is isolated and/or tackled. With a support license, you can use BSO/Tasking resources to solve your problem. You will get a direct access to experienced engineers. Furthermore, you receive (free of charge) the periodical *BSO/Tasking Newsletter*, regarding revisions, hints and documentation corrections.

Memory Models

MODEL	DATA ALLOCATION	APPLICATION
small	static, in on-chip direct addressable RAM (*data*)	fast programs in small environments
auxpage	static, in first page of external RAM (*pdat*)	derivatives with 256 bytes on-chip "external" RAM
large	static, in external RAM (*xdat*)	fast, non reentrant with large external RAM
reentrant	dynamic + static, in external RAM (*xdat*)	large, reentrant programs in large environments

ORDERING INFORMATION

TYPE NUMBER	DESCRIPTION
OM4136	8051 C-compiler package
Related Products	
OM4144	PLM-51 compiler
OM4142	8051 cross-assembler and linker
OM4129	XRAY51 high-level language debugger for the SDS 8051

Orders can be placed via your local Philips Semiconductors sales representative.

Sourced from BSO/Tasking Software B.V., Amersfoort, The Netherlands, which holds all intellectual property rights for this software.

™ Intel is a trademark of Intel Corporation.
MS-DOS and XENIX are trademarks of Microsoft Corporation.
XRAY is a trademark of Microtec Research Inc.
UNIX is a trademark of AT&T Bell Laboratories.
VAX, microVAX, VMS and Ultrix are trademarks of Digital Equipment Corp.
HP9000 and HP-UX are trademarks of Hewlett Packard Co.
SUN-3 and SunOS are trademarks of Sun Microsystems Inc.
IBM PC is a trademark of International Business Machines Corp.

Symbolic debugging package XRAY51 for the SDS 8051 emulator

OM4129

The XRAY™51 package enables the software developer to monitor and control the execution of a target program using the same high-level or assembly-level terms, definitions and structures found in the original source program. It employs a window-oriented user interface which segregates the debugging information into meaningful areas. Two versions of XRAY51, each with identical user interface, are available:

- An emulator version* using Philips' SDS 8051 emulation hardware as engine
- A simulator version* using a software engine.

The emulator (debugger) facilitates debugging in real-time on a real target; the simulator supports debugging in an 8051 environment, simulated in software on the host system. Both versions control the execution of the program and allow the user to stop, start, and examine the target software by means of a powerful command language that employs C language expressions.

XRAY51 HIGH-LEVEL DEBUGGER

The XRAY51 debugger helps to locate programming errors in the source code of C, PL/M or assembly language programs for the 8051 family of microcontrollers. The user can isolate these errors by controlling and monitoring program execution. For example, the user can single-step through the program a specified number of microcontroller instructions or high-level language lines.

Variables can be accessed with respect to the source language in which they had been defined (i.e., PL/M or Assembly). Operating XRAY51 in Assembly mode, the user can manipulate the contents of all processor registers. Only those registers that are part of the selected 8051 derivative are allowed to be accessed.

Macros may be defined that can execute complex user command procedures and provide a variety of complex breakpoints.

When debugging with XRAY51, the user can examine the contents and modify the value of any variable, compute the value of C, PL/M source language expressions and assembly level address expressions, and define, remove, or display symbols.

Command files can be used to direct XRAY51 to read or write simulated microprocessor input/output from or to a file, allowing easy implementation of automated test sequences. Command files enable scripts of debugger commands to be processed automatically without the need of user interaction.

XRAY51 also allows a virtually unlimited number of user-defined windows.

FEATURES

The XRAY51 debugger has been designed with the developer of embedded applications in mind. Its principal features are:

- Integrated PL/M-51 source-language and assembly-language debugging; toggle by function key at any time
- Window-oriented display which segregates debugging information into meaningful areas
- Symbolic debugging with C, PL/M-51 variables and C, PL/M-51 statements
- Simple and complex breakpoints
- Single-step execution
- User-definable screens and viewports—ability to write selected information
- Command macros
- Breakpoint macros (limited to the number of breakpoints of the SDS 8051)
- Command breakpoint macros may contain C statements, including: FOR, WHILE, DO, PRINTF, and FPRINTF
- On-line context-sensitive help
- Command files
- Output logging
- Fully supports SDS 8051 with the latest firmware version
- High-level trace
- Transparent ICE mode for direct control of SDS 8051
- Supports most 8051-derived microcontrollers, including the 80C51/31, 87C51, 8XC451/528/552/562/592/652/654/751/851, 8XCL410/710/51 and the 8051/31/52/32.

™ XRAY is a trademark of Microtec Research Inc., which holds all intellectual property rights for XRAY51.

* The SDS 8051 emulator version of XRAY51 is available from Philips Semiconductors (type number OM4129) and is sourced from BSO/Tasking Software B.V., Amersfoort, The Netherlands. The simulator version is available from BSO/Tasking Software B.V.

Symbolic debugging package XRAY51 for the SDS 8051 emulator

COMMANDS

Note, all commands can be issued in an abbreviated form.

Session Control Commands

HOST	Enter the host operating system environment
LOAD	Load an object module for debugging
QUIT	Terminate a debugging session

Execution and Breakpoint Commands

BREAKINSTRUCTION	Set an instruction breakpoint
CLEAR	Clear a breakpoint
GO	Start or continue program execution
GOSTEP	Execute macro after each instruction step
STEP	Execute a specified number of instruction lines
STEPOVER	Step, but execute through procedures

Display Commands

DISASSEMBLE	Display disassembled memory (assembly mode)
DUMP	Display memory contents
EXPAND	Display all local variables of a procedure
FIND	Search for a string
FOPEN	Open a file or device for writing
FPRINTF	Print formatted output to a viewport or file
LIST	Display source code
MONITOR	Monitor variables
NEXT	Find next occurrence of a string
NOMONITOR	Discontinue monitoring variables
PRINTF	Print formatted output to command viewport
PRINTVALUE	Print the value of a variable

Memory Commands

COMPARE	Compare two blocks of memory
COPY	Copy a memory block
FILL	Fill a memory block with values
SEARCH	Search a memory block for a value
SETMEM	Change the values of memory locations
SETREG	Change the contents of a register
TEST	Examine memory area for invalid values

Port I/O and Interrupt Commands

DIN	Display input port buffer values
DOUT	Display output port buffer values
INPORT	Set or alter input port status
INTERRUPT	Simulate an interrupt
NOINTERRUPT	Cancel pending interrupts
OUTPORT	Set or alter output port status
RIN	Rewind input file associated with input port
ROUT	Rewind output file associated with output port

Symbol Commands

ADD	Create a symbol
DELETE	Delete a symbol from the symbol table
PRINTSYMBOLS	Display symbol, type, and address
SCOPE	Specify current module and procedure scope

Utility Commands

CEXPRESSION	Calculate the value of an expression
ERROR	Set include-file error handling
HELP	Display On-line help screen
INCLUDE	Read in and process a command file
JOURNAL	Record a debugger session in a file
LOG	Record debugger commands and errors in a file
MODE	Select debugger mode (high or assembly)
OPTION	Set debugger options for this session
PAUSE	Pause simulation
RESET	Simulate microprocessor reset
RESTART	Restart program counter to the program starting address
STARTUP	Save the default start-up options

Macro Commands

DEFINE	Create a macro
SHOW	Display the macro source

Viewport Commands

VACTIVE	Activate a viewport
VCLEAR	Clear data from a viewport
VCLOSE	Remove a user-defined viewport or screen
VMACRO	Attach a macro to a viewport
VOPEN	Create a screen or viewport or change sizes
VSCREEN	Activate a screen
VSETC	Set the cursor position for a viewport
ZOOM	Increase or decrease the size of a viewport

Function Key Commands

VACTIVE–1	Activate the next viewport (counter clockwise)
VACTIVE+1	Activate the next viewport (clockwise)
MODE	Change debugging mode (assembly/high)
ZOOM	Increase or decrease the size of a viewport
HELP	Access on-line help
VSCREEN	Change the active screen
BACK UP	Back up one command
STEP	Execute one machine instruction or source line
STEP OVER	Step, but execute through procedures

In-Circuit Emulator Commands

ICE	Communicate with in-circuit emulator (ICE)
NOICE	Return to debugger command mode

RESTRICTIONS

- Up to 4 instruction breakpoints are allowed simultaneously.
- When an instruction has a breakpoint set, control is returned to XRAY *after* this instruction has been executed. However, the information about the break address returned is correct, so user actions (like macros) are executed as expected.
- In the current release, the TRACE capability of the SDS 8051 is supported by means of the ICE/NOICE command. The SDS 8051 TRACE information can be recorded in a journal file, by means of the JOURNAL command. The next release of XRAY51 will support emulator tracing within the debugger.

Symbolic debugging package XRAY51 for the SDS 8051 emulator

OM4129

HARDWARE/SOFTWARE REQUIREMENTS

XRAY51 software is supplied on $5^1/_4$" diskettes ($3^1/_2$" diskettes are available on request) together with the documentation: the *XRAY51 User's Guide, XRAY51 Installation Guide* and the *XRAY51 Reference Manual.* XRAY51 will run on an MS-DOS computer equipped with a hard disk and at least 512kbyte RAM.

To work with XRAY51,you will also need:

- An ASM51 Intel-compatible relocatable cross-assembler and LINK51 Intel-compatible linking loader (version 1.0 or later).

If you want to use the high-level language capabilities of XRAY51, the following PL/M-51 compiler is required:

- PLMTI51 (version 2.0 or later) with the ASM51 assembler

These products can be ordered from Philips Semiconductors. See Ordering Information.

Contact your local Philips representative for specific enquiries. Ask for the following leaflets:

Stand-alone debug station for 80C51/8051-based systems, ordering code 9398 366 00011;

PL/M-51 compiler package for 80C51/8051-based systems, ordering code 9398 366 20011;

Cross-assembler package for 80C51/8051-based systems, ordering code 9393 366 30011.

SOFTWARE MAINTENANCE AND SUPPORT

After a 90-day warranty period, in which all support will be given free of charge, a software update and support agreement can be taken out with Philips for a modest annual fee. Philips realizes your need for fast and comprehensive support, and with a support license, you get direct access to a development team that can solve your problems.

ORDERING INFORMATION

TYPE NUMBER	DESCRIPTION
OM4129	XRAY51 high-level language debugger for the SDS 8051; MS-DOS
Related Products	
OM4142	Cross-assembler package comprising the ASM51 assembler, and the LINK51 linker
OM4144	Compiler package comprising the PLMTI1

Orders can be placed via your local Philips Semiconductors sales representative.

™ Intel is a trademark of Intel Corporation.
MS-DOS is a trademark of Microsoft Corporation.

Section 7
Additional Microcontroller Data Sheets

80C51-Based 8-Bit Microcontrollers

CONTENTS

SCN8049 SERIES
SCN8049, SCN8050, SCN8039, SCN8040 Data Summary 911

8X305 Data Summary 912

8X401 Data Summary 913

PCA82C200 Data Sheet 914

UAA1300 Data Sheet 954

PCF1252-X Family Data Sheet 969

SCN8049, SCN8050, SCN8039, SCN8040 single-chip 8-bit microcontroller

SCN8049 SERIES

DESCRIPTION

The SCN8049 Series Microcontrollers are self-contained, 8-bit processors which contain the system timing, control logic, RAM data memory, ROM program memory (8048/49/50 only), and I/O lines necessary to implement dedicated control functions. All SCN8049 Series devices are pin and program compatible, differing only in the size of the on-board program ROM and data RAM, as follows:

TYPE	RAM SIZE	ROM SIZE
SCN8049	128 × 8	2k × 8
SCN8050	256 × 8	4k × 8
SCN8039	128 × 8	–
SCN8040	256 × 8	–

Program memory can be expanded externally up to a maximum total of 4k bytes without paging. Data memory can also be expanded externally. I/O capabilities can be expanded using standard devices or the 8243 I/O expander.

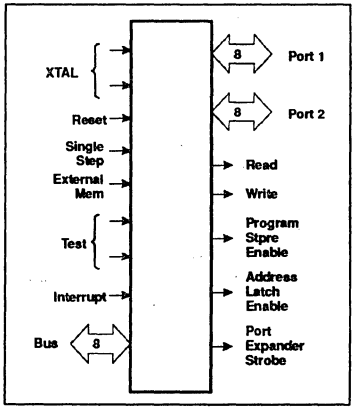
The SCN8049 Series processors are designed to be efficient control processors as well as arithmetic processors. They provide an instruction set which allows the user to directly set and reset individual lines within its I/O ports as well as test individual bits within the accumulator. A large variety of branch and table look-up instructions make these processors very efficient in implementing standard logic functions. Also, special attention has been given to code efficiency. Over 70% of the instructions are a single byte long and all others are only 2 bytes long.

An on-chip 8-bit counter is provided which can count, under program control, either internal clock pulses (with a divide by 32 prescaler) or external events. The counter can be programmed to cause an interrupt on terminal count.

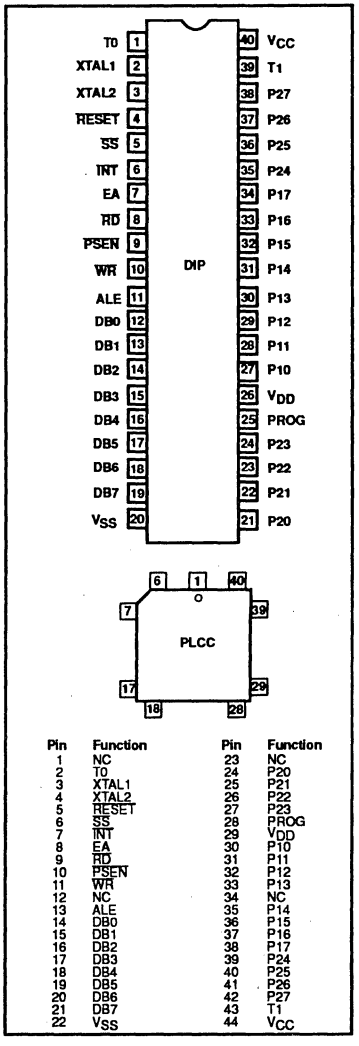
FEATURES

- 8-bit CPU, ROM, RAM, I/O in a 40-pin package
- 24 quasi-bidirectional I/O lines
- Two test inputs
- Internal counter/timer
- Single-level vectored interrupts: external, counter/timer
- Over 90 instructions, 70% single byte
- 1.36μs or 2.5μs instruction cycle, all instructions one or two cycles
- Expandable memory and I/O
- Low voltage standby
- TTL compatible inputs and outputs
- Single +5V power supply

LOGIC SYMBOL



PIN CONFIGURATIONS



Pin	Function	Pin	Function
1	NC	23	NC
2	T0	24	P20
3	XTAL1	25	P21
4	XTAL2	26	P22
5	RESET	27	P23
6	SS	28	PROG
7	INT	29	VDD
8	EA	30	P10
9	RD	31	P11
10	PSEN	32	P12
11	WR	33	P13
12	NC	34	NC
13	ALE	35	P14
14	DB0	36	P15
15	DB1	37	P16
16	DB2	38	P17
17	DB3	39	P24
18	DB4	40	P25
19	DB5	41	P26
20	DB6	42	P27
21	DB7	43	T1
22	VSS	44	VCC

FOR COMPLETE INFORMATION ON THIS PRODUCT, CONTACT YOUR LOCAL SIGNETICS SALES OFFICE (SEE SECTION 9 OF THIS DATA HANDBOOK).

Microcontroller 8X305

FEATURES

- Fetch, Decode, and Execute a 16-bit instruction in a minimum of 200ns (one machine cycle)
- Bit-oriented instruction set (addressable single-or-multiple bit subfields)
- Separate buses for Instruction, Instruction Address and Three-State I/O
- Thirteen 8-bit general-purpose working registers
- Source/destination architecture
- Bipolar low-power Schottky technology/TTL inputs and outputs
- On-chip oscillator and timing generation
- Single +5V supply
- 0.9-in. 50-pin DIP
- 68-pin PLCC

DESCRIPTION

The 8X305 Microcontroller (Figure 1) is a high-speed bipolar microprocessor implemented with low-power Schottky technology. In a single chip, the 8X305 combines speed, flexibility, and a bit-oriented instruction set. These features and other basic characteristics of the chip combine to provide cost-effective solutions for a broad range of applications. The 8X305 is particularly useful in systems that require high-speed bit manipulations—sophisticated controllers, data communications, very fast interface control, and other applications of a similar nature.

The 8X305 can fetch, decode, and execute a 16-bit instruction in a minimum of 200ns. Within one instruction cycle, the 8-bit data-processing path can be programmed to rotate, mask, shift, and/or merge single or multiple bit subfields and, in addition, perform an ALU operation. In the same instruction, an external data field can be input, processed, and output to a specified destination—likewise, single or multiple bit data fields can be internally moved from a given source to a given destination. To summarize, fixed or variable-length data fields can be fetched, processed, operated on by the ALU, and moved to a different location—all in a timeframe of 200ns. To interface with I/O and program memory, the 8X305 uses a 13-bit instruction address bus, a 16-bit instruction bus, an 8-bit bidirectional multiplexed I/O data/address bus and a 5-bit I/O control bus.

A wide selection of I/O devices, interface chips, and special-purpose parts are available for systems use. In most applications, the more powerful 8X305 is functionally interchangeable with its predecessor—the 8X300.

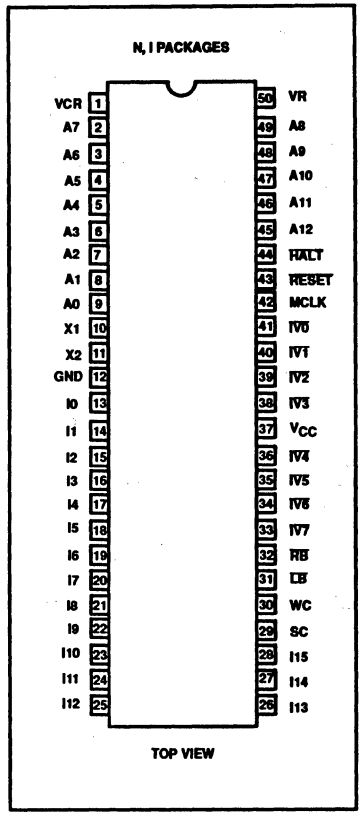
ASSOCIATED DOCUMENTATION

Other documents directly relating to *design* and *applications use* of the 8X305 Microcontroller are:

- Product Capabilities Manual
- 8X305 Users Manual

These documents and other current literature (Data Sheets, Product Bulletins, Applications Notes, etc.) are available from you local Signetics Sales Office.

PIN CONFIGURATION



ORDERING INFORMATION

DESCRIPTION	ORDER CODE
50-Pin plastic DIP	N8X305N
50-Pin ceramic DIP	N8X305I
68-Pin PLCC	N8X305A

FOR COMPLETE INFORMATION ON THIS PRODUCT, CONTACT YOUR LOCAL SIGNETICS SALES OFFICE (SEE SECTION 9 OF THIS DATA HANDBOOK).

Microcontroller 8X401

DESCRIPTION

The 8X401 Microcontroller is a very high-speed bipolar microprocessor implemented with internal ECL technology. The 8X401 combines speed, flexibility, and a bit-oriented instruction set to accommodate many sophisticated applications. It excels in systems that require high-speed bit or byte manipulations, such as high-speed controllers and data communications.

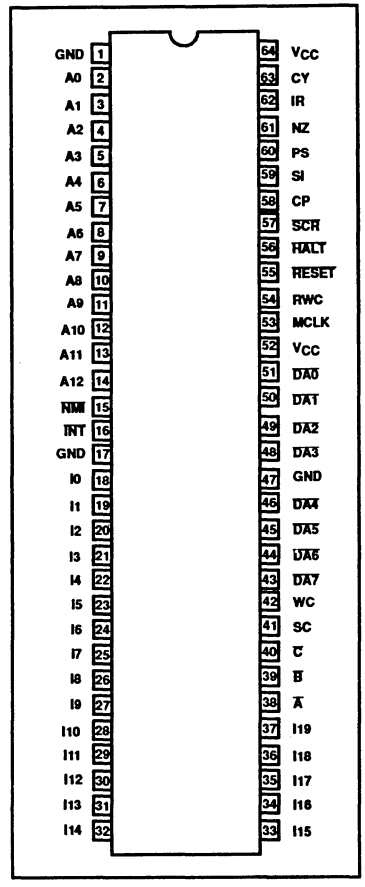
The 8X401 can fetch, execute, and generate the next instruction address for a 20-bit instruction in a minimum of 150ns. Within one instruction cycle, the 8X401 can be programmed to input, right-rotate and mask single or multiple bit subfields, perform an ALU operation, left-rotate, merge the subfield into the destination, and output.

To interface with program memory, the 8X401 uses a 13-bit address bus and a 20-bit instruction bus. An 8-bit bidirectional data/address bus, and an I/O control and timing bus is used to access external peripheral devices.

FEATURES

- Fetches and executes all instructions in a minimum of 150ns
- Bit manipulation-oriented instruction set
- Separate buses for instruction, instruction address and I/O
- Sixteen 8-bit registers
- On-chip interrupt control
- TTL compatible I/O
- Single +5V supply
- 0.9-inch 64-pin DIP, 68-pin plastic leaded chip carrier
- Two user-definable status flags
- Single TTL clock input
- Three independent I/O banks
- On-board control sequencer
- On-chip subroutine capabilities
- Fixed instruction set—32 instructions
- Complete development support

PIN CONFIGURATION



ORDERING INFORMATION

DESCRIPTION	ORDER CODE
64-Pin Ceramic DIP – 900 Mil Wide	N8X401I
64-Pin Ceramic DIP – 900 Mil Wide	N8X401IN

FOR COMPLETE INFORMATION ON THIS PRODUCT, CONTACT YOUR LOCAL SIGNETICS SALES OFFICE (SEE SECTION 9 OF THIS DATA HANDBOOK).

Stand-alone CAN-controller PCA82C200

1.0 FEATURES

- Multi-master architecture
- Interfaces with a large variety of microcontrollers (Intel, Motorola or Intel and Motorola compatible)
- Bus access priority (determined by the message identifier)
- 2032 message identifiers
- Guaranteed latency time for high priority messages
- Powerful error handling capability
- Data length from 0 to 8 bytes
- Configurable bus interface
- Programmable clock output
- Multicast and Broadcast message facility
- Non destructive bit-wise arbitration
- Non-return-to-zero (NRZ) coding/decoding with bit-stuffing
- Programmable transfer rate (up to 1 Mbit/s)
- Programmable output driver configuration
- Suitable for use in a wide range of networks including the SAE networks Class A, B and C
- 16 MHz clock frequency
- −40 to + 125 °C operating temperature

2.0 GENERAL DESCRIPTION

The PCA82C200 is a highly integrated stand-alone controller for the controller area network (CAN) used within automotive and general industrial environments. The PCA82C200 contains all necessary features required to implement a high performance communication protocol. The PCA82C200 with a simple bus line connection performs all the functions of the physical and data-link layers. The application layer of an Electronic Control Unit (ECU) is provided by a microcontroller, to which the PCA82C200 provides a versatile interface. The use of the PCA82C200 in an automotive or general industrial environment, results in a reduced wiring harness and an enhanced diagnostic and supervisory capability.

3.0 ORDERING INFORMATION

EXTENDED TYPE NUMBER	PACKAGE			
	PINS	PIN POSITION	MATERIAL	CODE
PCA82C200P	28	DIL	plastic	SOT117
PCA82C200T	28	SO28	plastic	SOT136A

CONTENTS

1.0 FEATURES

2.0 GENERAL DESCRIPTION

3.0 ORDERING INFORMATION

4.0 PINNING INFORMATION
4.1 Pinning

5.0 DEVELOPMENT SUPPORT AND TOOLS
5.1 The PCA82C200 evaluation board
5.2 Advanced support

6.0 FUNCTIONAL DESCRIPTION
6.1 Interface Management Logic (IML)
6.2 Transmit Buffer (TBF)
6.3 Receive Buffers (RBF0 and RBF1)
6.4 Bit Stream Processor (BSP)
6.5 Bit Timing Logic (BTL)
6.6 Transceiver Control Logic (TCL)
6.7 Error Management Logic (EML)
6.8 Controller Interface Logic (CIL)

7.0 CONTROL SEGMENT AND MESSAGE BUFFER DESCRIPTION
7.1 Address allocation
7.2 Control Segment layout
7.2.1 Control Register (CR)
7.2.2 Command Register (CMR)
7.2.3 Status Register (SR)
7.2.4 Interrupt Register (IR)
7.2.5 Acceptance Code Register (ACR)
7.2.6 Acceptance Mask Register (AMR)
7.2.7 Bus Timing Register 0 (BTR 0)
7.2.8 Bus Timing Register 1 (BTR 1)
7.2.9 Output Control Register (OCR)
7.2.10 Test Register (TR)
7.3 Transmit Buffer layout
7.3.1 Descriptor
7.3.2 Data Field
7.4 Receive Buffer layout
7.5 Clock Divider Register (CDR)

8.0 BUS TIMING/SYNCHRONIZATION
8.1 Bit timing
8.1.1 Synchronization Segment (SYNCSEG)
8.1.2 Time Segment 1 (TSEG1)
8.1.3 Time Segment 2 (TSEG2)
8.1.4 Synchronization Jump Width (SJW)
8.1.5 Propagation delay time
8.1.6 Bit timing restrictions
8.2 Synchronization
8.2.1 Hard synchronization
8.2.2 Resynchronization
8.2.3 Synchronization rules

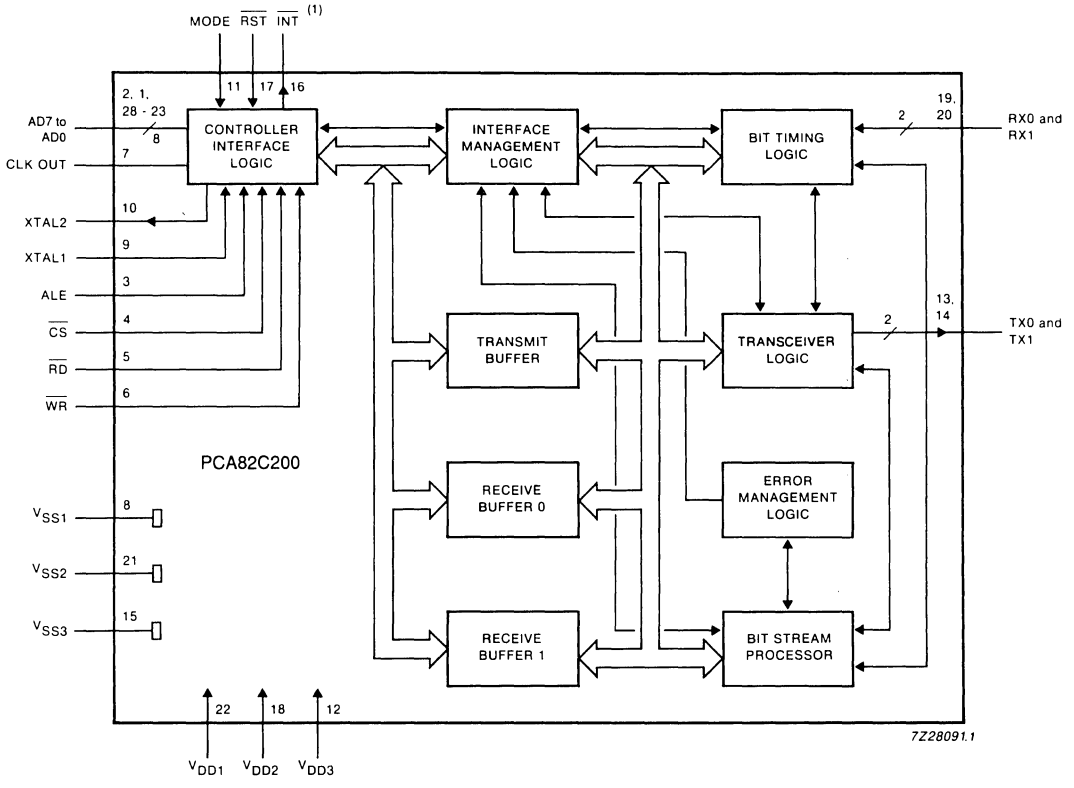
9.0 COMMUNICATION PROTOCOL
9.1 Frame types
9.1.1 Bit representation
9.2 Data Frame
9.2.1 Start-of-Frame
9.2.2 Arbitration Field
9.2.3 Control Field
9.2.4 Data Field
9.2.5 Cyclic Redundancy Code (CRC) Field
9.2.6 Acknowledge Field
9.2.7 End-of-Frame Field
9.3 Remote Frame
9.4 Error Frame
9.4.1 Error Flag
9.4.2 Error Delimiter
9.5 Overload Frame
9.5.1 Overload Flag
9.5.2 Overload Delimiter
9.6 Inter-Frame Space
9.6.1 Intermission Field
9.6.2 Bus-Idle
9.7 Bus organization
9.7.1 Bus access
9.7.2 Arbitration
9.7.3 Coding/decoding
9.7.4 Error signalling
9.7.5 Overload signalling
9.8 Error detection
9.8.1 Bit Error

9.8.2 Stuff Error
9.8.3 CRC Error
9.8.4 Form Error
9.8.5 Acknowledgement error
9.8.6 Error detection by an Error Flag of another PCA82C200
9.8.7 Error detection capabilities
9.9 Error confinement (definitions)
9.9.1 Bus-Off
9.9.2 Acknowledge (ACK)
9.9.3 Error-Active
9.9.4 Error-Passive
9.9.5 Suspend Transmission
9.9.6 Start-up
9.10 Aims of error confinement
9.10.1 Distinction of short and long lasting disturbances
9.10.2 Detection and localization of hardware disturbances and defects
9.10.3 Error confinement

10.0 LIMITING VALUES

11.0 DC CHARACTERISTICS

12.0 AC CHARACTERISTICS
12.1 AC timing diagrams
12.2 Additional AC information



Note to Fig.2.1

(1) During 'sleep mode' this pin functions as an input.

Fig.2.1 Block diagram.

4.0 PINNING INFORMATION

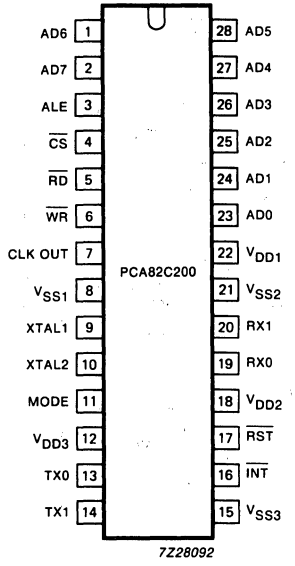


Fig.4.1 Pin configuration.

4.1 Pinning

SYMBOL	PIN	DESCRIPTION
AD7 - AD0	2, 1, 28 - 23	Multiplexed address/data bus.
ALE	3	ALE signal (Intel mode) or AS input signal (Motorola mode).
$\overline{CS}$	4	Chip select input, low level allows access to the PCA82C200.
$\overline{RD}$	5	$\overline{RD}$ signal (Intel mode) or E enable signal (Motorola mode) from the microcontroller.
$\overline{WR}$	6	$\overline{WR}$ signal (Intel mode) or RD/$\overline{WR}$ signal (Motorola mode) from the microcontroller.
CLK OUT	7	Clock output signal produced by the PCA82C200 for the microcontroller. The clock signal is derived from the built-in oscillator, via the programmable divider (see section 7.5). This output is capable of driving one CMOS or NMOS load.
V_{SS1}	8	Ground potential for the logic circuits.
XTAL1 (note 1)	9	Input to the oscillator's amplifier. External oscillator signal is input via this pin.
XTAL2 (note 1)	10	Output from the oscillator's amplifier. Output must be left open when an external oscillator signal is used.
MODE	11	Mode select input: connected to V_{DD} selects Intel mode; connected to V_{SS} selects Motorola mode.
V_{DD3}	12	5 V power supply for the output driver.
TX0	13	Output from the output-driver 0 to the physical bus-line.
TX1	14	Output from the output-driver 1 to the physical bus-line.
V_{SS3}	15	Ground potential for the output driver.
$\overline{INT}$	16	Interrupt output, used to interrupt the microcontroller (see section 7.2.4). $\overline{INT}$ is active if the Interrupt Register contains a logic HIGH bit (present). $\overline{INT}$ is an open drain output and is designed to be a wired-OR with other $\overline{INT}$ outputs within the system. A LOW level on this pin will reactivate the IC from the sleep mode (see section 7.2.2).
$\overline{RST}$	17	Reset input, used to reset the CAN interface (LOW level). Automatic power-ON reset can be obtained by connecting $\overline{RST}$ via a capacitor to V_{SS} and via a resistor to V_{DD} (e.g. C = 1 μF; R = 50 kΩ).
V_{DD2}	18	5 V power supply for the input comparator.
RX0 - RX1	19, 20	Input from the physical bus-line to the input comparator of the PCA82C200. A dominant level will wake-up the PCA82C200. A recessive level is read if RX0 is higher than RX1 and vice versa for the dominant level.
V_{SS2}	21	Ground potential for the input comparator.
V_{DD1}	22	5 V power supply for the logic circuits.

Note

1. XTAL1 and XTAL2 pins should be connected to V_{SS} via 15 pF capacitors.

5.0 DEVELOPMENT SUPPORT AND TOOLS

5.1 The PCA82C200 evaluation board

Philips offers powerful support during the design and test stages of CAN networks, working closely with customers to develop their systems. The "Philips Stand-alone CAN Controller (PSCC) Evaluation Board" is a versatile tool being a ready-to-use hardware and software module, very similar to a real CAN module. Since a 5 V power supply is provided, the board can be used in any vehicle without modification. An RS232 interface allows a terminal or a PC with terminal-emulation software to be connected to the board. The board comprises:

- a PCA82C200 CAN bus-controller
- a PCA80C552 microcontroller with up to 32K x 8 bits external RAM and EPROM
- a 5 V power supply with protection against car battery disturbances
- two different physical CAN bus interfaces (selectable)
- an RS232 interface
- demonstration hardware
- a wrap field for customer-specific circuitry

The software provided with the board supports in learning about CAN and assists in prototyping (e.g. in-vehicle) networks. It provides:

- demonstration software (automatically startable)
- the menu-driven software comprises:
  - a facility to alter the contents of the PCA82C200 registers
  - a bus monitor to receive messages from the CAN bus and to display them on a terminal
  - a download facility for the user's application software

With these facilities the board is a basis for prototype modules; using entirely your own software, the board can be used as a custom, debugged and proven hardware module.

5.2 Advanced support

For further development support, Philips subcontractor I+ME offers a complete set of development tools including:

- a CAN simulator; CAN/Net Sim
- an emulator; CAN/Net Emu
- a network analyzer; CAN/Net Anal

I+ME can be contacted through the following address:

I+ME GmbH
Ferdinandstrasse 15
D-3340 Wolfenbuettel
West Germany.

6.0 FUNCTIONAL DESCRIPTION

The PCA82C200 contains all necessary hardware for a high performance serial network communication (see Fig.2.1). The PCA82C200 controls the communication flow through the area network using the CAN-protocol. The PCA82C200 meets the following automotive requirements:

- short message length
- guaranteed latency time for urgent messages
- bus access priority, determined by the message identifier
- powerful error handling capability
- configuration flexibility to allow area network expansion.

The latency time defines the period between the initiation (Transmission Request) and the start of the transmission on the bus. Latency time is dependent on a variety of bus related conditions. In the case of a message being transmitted on the bus and one distortion the latency time can be up to 149 bit times (worst case). For more information see application note on 'Bit Timing'.

6.1 Interface Management Logic (IML)

The IML interprets commands from the microcontroller, allocates the message buffers (TBF, RBF0 and RBF1) and provides interrupts and status information to the microcontroller.

6.2 Transmit Buffer (TBF)

The TBF is a 10 byte memory into which the microcontroller writes messages which are to be transmitted over the CAN network.

6.3 Receive Buffers (RBF0 and RBF1)

The RBF0 and RBF1 are each 10 byte memories which are alternatively used to store messages received from the CAN network. The CPU can process one message while another is being received.

6.4 Bit Stream Processor (BSP)

The BSP is a sequencer, controlling the data stream between the Transmit Buffer, Receiver Buffers (parallel data) and the CAN bus (serial data).

6.5 Bit Timing Logic (BTL)

The BTL synchronizes the PCA82C200 to the bitstream on the CAN bus.

6.6 Transceiver Control Logic (TCL)

The TCL controls the output driver.

6.7 Error Management Logic (EML)

The EML performs the error confinement according to the CAN protocol.

6.8 Controller Interface Logic (CIL)

The CIL is the interface to the external microcontroller. The PCA82C200 can directly interface with a variety of microcontrollers.

7.0 CONTROL SEGMENT AND MESSAGE BUFFER DESCRIPTION

The PCA82C200 appears to a microcontroller as a memory-mapped I/O device due to the on-chip RAM, guaranteeing the independent operation of both devices.

7.1 Address allocation

The address area of the PCA82C200 consists of the Control Segment and the message buffers. The Control Segment is programmed during an initialization download in order to configure communication parameters (e.g. bit timing). Communication over the CAN-bus is also controlled via this segment by the microcontroller. During initialization the CLOCK OUT signal may be programmed to a value determined by the microcontroller (see Fig.2.1). A message which is to be transmitted, must be written to the Transmit Buffer. After a successful reception the microcontroller may read the message from the Receive Buffer and then release it for further use.

7.2 Control Segment layout

The exchange of status, control and command signals between the microcontroller and the PCA82C200 is performed in the control segment. The layout of this segment is shown in Fig.7.1. After an initial down-load, the contents of the registers Acceptance Code, Acceptance Mask, Bus Timing 0 and 1, and Output Control should not be changed. These registers may only be accessed when the Reset Request bit in the Control Register, is set HIGH (see section 7.2.1).

Table 7.1 Register map.

DESCRIPTION	AD-DRESS	7 (MSB)	6	5	4	3	2	1	0 (LSB)
Control Segment									
Control Register	0	Test Mode	Synch	reserved	Overrun Interrupt Enable	Error Interrupt Enable	Transmit Interrupt Enable	Receive Interrupt Enable	Reset Request
Command Register	1	reserved	reserved	reserved	Goto Sleep	Clear Overrun Status	Release Receive Buffer	Abort Transmission	Transmission Request
Status Register	2	Bus Status	Error Status	Transmit Status	Receive Status	Transmission Complete Status	Transmit Buffer Access	Data Overrun	Receive Buffer Status
Interrupt Register	3	reserved	reserved	reserved	Wake-Up Interrupt	Overrun Interrupt	Error Interrupt	Transmit Interrupt	Receive Interrupt
Acceptance Code Register	4	AC.7	AC.6	AC.5	AC.4	AC.3	AC.2	AC.1	AC.0
Acceptance Mask Register	5	AM.7	AM.6	AM.5	AM.4	AM.3	AM.2	AM.1	AM.0
Bus Timing Register 0	6	SJW.1	SJW.0	BRP.5	BRP.4	BRP.3	BRP.2	BRP.1	BRP.0
Bus Timing Register 1	7	SAM	TSEG2.2	TSEG2.1	TESG2.0	TSEG1.3	TSEG1.2	TSEG1.1	TSEG1.0
Output Control Register	8	OCTP1	OCTN1	OCPOL1	OCTP0	OCTN0	OCPOL0	OCMODE1	OCMODE0
Test Register (see note 1)	9	reserved	reserved	Map Internal Register	Connect RX Buffer 0 CPU	Connect TX Buffer CPU	Access Internal Bus	Normal RAM Connect	Float Output Driver
Transmit Buffer									
Identifier	10	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3
RTR Data Length Code	11	ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0
bytes 1 - 8	12 - 19	Data	Data	Data	Data	Data	Data	Data	Data
Receive Buffer 0/1									
Identifier	20	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3
RTR Data Length Code	21	ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0
bytes 1 - 8	22 - 29	Data	Data	Data	Data	Data	Data	Data	Data
Clock Divider	31	reserved	reserved	reserved	reserved	reserved	CD.2	CD.1	CD.0

Notes to Table 7.1

1. The Test Register is used for production testing only.
2. Register 30 is not implemented.

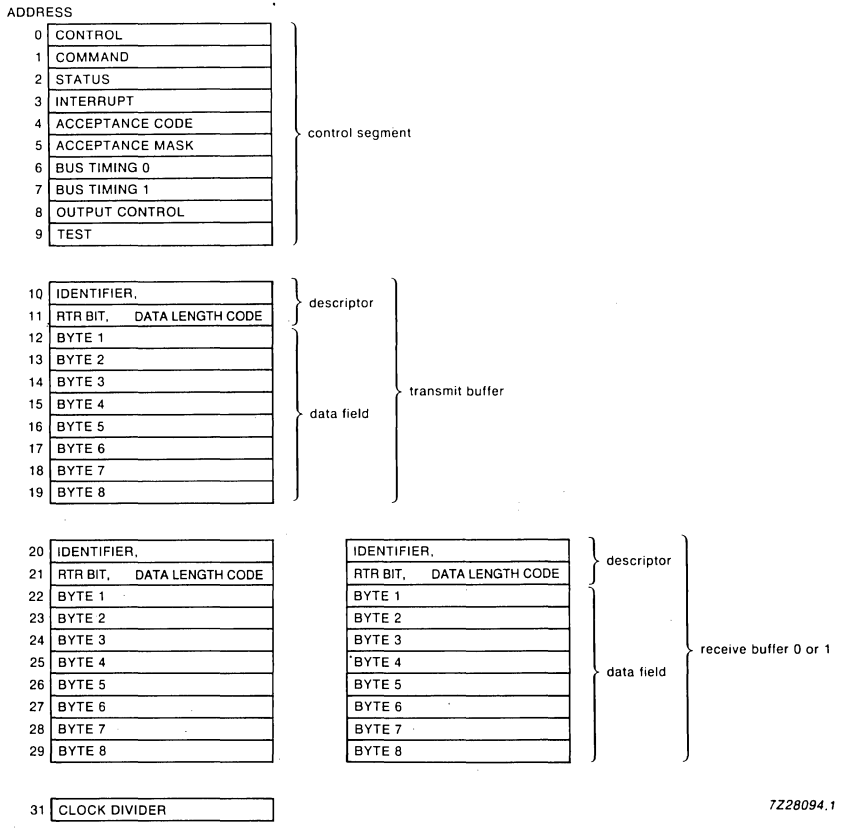


Fig 7.1 PCA82C200 address allocation.

7.2.1 Control Register (CR)

The contents of the Control Register are used to change the behaviour of the PCA82C200. Control bits may be set or reset by the attached microcontroller which uses the Control Register as a read/write memory.

Table 7.2 Control Register bits.

CR : ADDRESS 0		
BIT	**SYMBOL**	**FUNCTION**
CR.7	TM	Test Mode
CR.6	S	Synch
CR.5	-	Reserved
CR.4	OIE	Overrun Interrupt Enable
CR.3	EIE	Error Interrupt Enable
CR.2	TIE	Transmit Interrupt Enable
CR.1	RIE	Receive Interrupt Enable
CR.0	RR	Reset Request

Table 7.3 Description of the Control Register bits.

CONTROL BIT	VALUE	COMMENTS
Reset Request (note 1)	HIGH (present)	Detection of a Reset Request results in the PCA82C200 aborting the current transmission\reception of a message and entering the reset state.
	LOW (absent)	On the HIGH-to-LOW transition of the Reset Request bit, the PCA82C200 returns to its normal operating state.
Receive Interrupt Enable	HIGH (enabled)	When a message has been received without errors, the PCA82C200 transmits a Receive Interrupt signal to the microcontroller.
	LOW (disabled)	No transmission of the Receive Interrupt signal by the PCA82C200 to the microcontroller.
Transmit Interrupt Enable	HIGH (enabled)	When a message has been successfully transmitted or the transmit buffer is accessible again, (e.g. after an Abort Transmission command) the PCA82C200 transmits a Transmit Interrupt signal to the microcontroller.
	LOW (disabled)	No transmission of the Transmit Interrupt signal by the PCA82C200 to the microcontroller.
Error Interrupt Enable	HIGH (enabled)	If the Error or Bus Status change (see section 7.2.3), the microcontroller receives an Error Interrupt signal.
	LOW (disabled)	Microcontroller receives no Error Interrupt signal.
Overrun Interrupt Enable	HIGH (enabled)	If the Data Overrun bit is set (see section 7.2.3), the microcontroller receives an Overrun Interrupt signal.
	LOW (disabled)	Microcontroller receives no Overrun Interrupt signal from the PCA82C200.
Synch (note 2)	HIGH (2 edges)	Bus-line transitions from recessive-to-dominant and vice versa are used for resynchronization (see sections 8.2 and 9.0).
	LOW (1 edge)	Only transitions from recessive-to-dominant are used for resynchronization.
Test Mode (note 3)	HIGH (enabled) LOW (disabled)	PCA82C200 enters Test Mode (normal operation impossible). Normal operating mode. Note: if Reset Request bit = HIGH, the Test Mode bit is set LOW.

Notes to Table 7.3

1. During an external reset ($\overline{RST}$ = LOW) or when the Bus Status bit is set HIGH (Bus-Off), the IML forces the Reset Request HIGH (present). During an external reset the microcontroller cannot set the Reset Request bit LOW (absent). Therefore, after having set the Reset Request bit LOW (absent), the microcontroller must check this bit to ensure that the external reset pin is not being held HIGH (present). After the Reset Request bit is set LOW (absent) the PCA82C200 will wait for:
   - one occurrence of the Bus-Free signal (11 recessive bits, see section 9.9.6), if the preceding reset (Reset Request = HIGH) was due to an external reset or a microcontroller initiated reset
   - 128 occurrences of Bus-Free, if the preceding reset (Reset Request = HIGH) was due to a PCA82C200 initiated Bus-Off, before re-entering the Bus-On mode (see section 9.9).

   When Reset Request is set HIGH (present), for whatever reason, the control, command, status and interrupt bits are affected, see Table 7.4.
   When Reset Request is set HIGH (present) the registers at addresses 4 to 8 are accessible but the TBF is not.
2. The Synch bit should only be modified if the Reset Request bit is set HIGH (present), otherwise it is ignored. It is possible to set the Synch bit while the Reset Request bit is changed from HIGH to LOW.
3. The Test mode is intended for factory testing and not for customer use.

Table 7.4 Effects of setting the Reset Request bit HIGH (present).

TYPE	BIT	EFFECT
Control	Test Mode	LOW (disabled)
Command	Goto Sleep	LOW (wake-up)
	Clear Overrun Status	HIGH (clear)
	Release Receive Buffer	HIGH (released)
	Abort Transmission	LOW (absent)
	Transmission Request	LOW (absent)
Status	Bus Status	LOW (Bus-On), see note 1
	Error Status	LOW (no error), see note 1
	Transmit Status	LOW (idle)
	Receive Status	LOW (idle)
	Transmission Complete Status	HIGH (complete)
	Transmit Buffer Access	HIGH (released)
	Data Overrun	LOW (absent)
	Receive Buffer Status	LOW (empty)
Interrupt	Overrun Interrupt	LOW (reset)
	Transmit Interrupt	LOW (reset)
	Receive Interrupt	LOW (reset)

Note to Table 7.4

1. Only after an external reset; see note 5 to Table 7.8.

7.2.2 Command Register (CMR)

A command bit initiates an action within the transfer layer of the PCA82C200. The Command Register appears to the microcontroller as a write only memory. If a read access is performed to this address the byte 11111111_b is returned.

Table 7.5 Command Register bits.

CMR : ADDRESS 1		
BIT	**SYMBOL**	**FUNCTION**
CMR.7	-	Reserved
CMR.6	-	Reserved
CMR.5	-	Reserved
CMR.4	GTS	Goto Sleep
CMR.3	COS	Clear Overrun Status
CMR.2	RRB	Release Receive Buffer
CMR.1	AT	Abort Transmission
CMR.0	TR	Transmission Request

Table 7.6 Description of the Command Register bits.

CONTROL BIT	VALUE	COMMENTS
Transmission Request (note 1)	HIGH (present) LOW (absent)	A message shall be transmitted. No action.
Abort Transmission (note 2)	HIGH (present) LOW (absent)	If not already in progress, a pending Transmission Request is cancelled. No action.
Release Receive Buffer (note 3)	HIGH (released) LOW (no action)	The Receive Buffer attached to the microcontroller is released. No action.
Clear Overrun (note 4)	HIGH (clear) LOW (no action)	The Data Overrun status bit is set to LOW (see section 7.2.3). No action.
Goto Sleep (note 5)	HIGH (sleep) LOW (wake up)	The PCA82C200 enters sleep mode, if the $\overline{INT}$ = HIGH (no interrupt signal from the PCA82C200 to the microcontroller pending or external source pending) and there is no bus activity. The PCA82C200 functions normally.

Notes to Table 7.6

1. If the Transmission Request bit was set HIGH in a previous command, it cannot be cancelled by setting the Transmission Request bit LOW (absent). Cancellation of the requested transmission may be performed by setting the Abort Transmission bit HIGH (present).
2. The Abort Transmission bit is used when the microcontroller requires the suspension of the previously requested transmission, for example to transmit an urgent message. A transmission already in progress is not stopped. In order to determine if the original message had been transmitted successfully, or aborted, the Transmission Complete status bit should be checked after the Transmit Buffer Access bit has been set HIGH (released) or a Transmit Interrupt has been generated (see section 7.2.4).
3. After reading the contents of the Receive Buffer (RBF0 or RBF1) the microcontroller must release this buffer by setting the Release Receive Buffer bit HIGH (released). This may result in another message becoming immediately available.
4. This command bit is used to acknowledge the Data Overrun condition signalled by the Data Overrun status bit. It may be given or set at the same time as a Release Receive Buffer command bit.
5. The PCA82C200 will enter sleep mode, if Goto Sleep is set HIGH (sleep), there is no bus activity and $\overline{INT}$ = HIGH (inactive). After sleep mode is set, the CLK OUT signal continues until at least 15 bit times have passed. The PCA82C200 will wake up when one of the three previously mentioned conditions is negated: after Goto Sleep is set LOW (wake up), there is bus activity or $\overline{INT}$ is driven LOW (active). On wake up, the oscillator is started and a Wake-Up Interrupt (see section 7.2.4) is generated. A PCA82C200 which is sleeping and then awoken by bus activity will not be able to receive this message until it detects a Bus-Free signal (see section 9.9.6).

7.2.3 Status Register (SR)

The contents of the Status Register reflect the status of the PCA82C200 bus controller. The Status Register appears to the microcontroller as a read only memory.

Table 7.7 Status Register bits.

SR : ADDRESS 2		
BIT	**SYMBOL**	**FUNCTION**
SR.7	BS	Bus Status
SR.6	ES	Error Status
SR.5	TS	Transmit Status
SR.4	RS	Receive Status
SR.3	TCS	Transmission Complete Status
SR.2	TBS	Transmit Buffer Status
SR.1	DO	Data Overrun
SR.0	RBS	Receive Buffer Status

Table 7.8 Description of the Status Register bits.

STATUS BIT	VALUE	COMMENTS
Receive Buffer Status (note 1)	HIGH (full) LOW (empty)	This bit is set when a new message is available. No message has become available since the last Release Receive Buffer command bit was set.
Data Overrun (note 2)	HIGH (overrun) LOW (absent)	This bit is set HIGH (Overrun), when both Receive Buffers are full and the first byte of another message should be stored. No data overrun has occurred since the Clear Overrun command was given.
Transmit Buffer Access(note 3)	HIGH (released) LOW (locked)	The microcontroller may write a message into the TBF. The microcontroller cannot access the Transmit Buffer. A message is either waiting for transmission or is in the process of being transmitted.
Transmission Complete Status (note 3)	HIGH (complete) LOW (incomplete)	Last requested transmission has been successfully completed. Previously requested transmission is not yet completed.
Receive Status (note 4)	HIGH (receive) LOW (idle)	The PCA82C200 is receiving a message. No message is received.
Transmit Status (note 4)	HIGH (transmit) LOW (idle)	The PCA82C200 is transmitting a message. No message is transmitted.
Error Status	HIGH (error) LOW (ok)	At least one of the Error Counters (see 9.10.3) has reached the microcontroller Warning Limit. Both Error Counters have not reached the Warning Limit.
Bus Status (note 5)	HIGH (Bus-Off) LOW (Bus-On)	The PCA82C200 is not involved in bus activities. The PCA82C200 is involved in bus activities.

Notes to Table 7.8

1. If the command bit Release Receive Buffer is set HIGH (released) by the microcontroller, the Receive Buffer Status bit is set LOW (empty) by IML. When a new message is stored in any of the receive buffers, the Receive Buffer Status bit is set HIGH (full) again.
2. If Data Overrun = HIGH (Overrun) is detected, the currently received message is dropped.
   A transmitted message, granted acceptance, is also stored in a Receive Buffer. This occurs because it is not known if the PCA82C200 will lose arbitration and so become a receiver of the message. If no receive buffer is available, Data Overrun is signalled.
3. If the microcontroller tries to write to the Transmit Buffer when the Transmit Buffer Access bit is LOW (locked), the written bytes will not be accepted and will be lost without this being signalled.
   The Transmission Complete Status bit is set LOW (incomplete) whenever the Transmission Request bit is set HIGH (present). If an Abort Transmission command is issued, the Transmit Buffer will be released. If the message, which was requested and then aborted, was not transmitted, the Transmission Complete Status bit will remain LOW.
4. If both the Receive Status and Transmit Status bits are LOW (idle) the CAN-bus is idle.
5. When the Bus Status bit is set HIGH (Bus-Off), the PCA82C200 will set the Reset Request bit HIGH (present). It will stay in this state until the microcontroller sets the Reset Request bit LOW (absent). Once this is completed the PCA82C200 will wait the minimum protocol-defined time (128 occurrences of the Bus-Free signal) before setting the Bus Status bit LOW (Bus-On), the Error Status bit LOW (OK) and resetting the Error Counters.

7.2.4 Interrupt Register (IR)

The Interrupt Register allows the identification of an interrupt source. When one or more bits of this register are set, the $\overline{INT}$ pin is activated. All bits are reset by the PCA82C200 after this register is read by the microcontroller. This register appears to the microcontroller as a read only memory.

Table 7.9 Interrupt Register bits.

IR : ADDRESS 3		
BIT	**SYMBOL**	**FUNCTION**
IR.7	-	Reserved
IR.6	-	Reserved
IR.5	-	Reserved
IR.4	WUI	Wake-Up Interrupt
IR.3	OI	Overrun Interrupt
IR.2	EI	Error Interrupt
IR.1	TI	Transmit Interrupt
IR.0	RI	Receive Interrupt

Table 7.10 Description of the Interrupt Register bits.

INTERRUPT BIT	VALUE	COMMENTS
Receive Interrupt (note 1)	HIGH (set)	This bit is set when a new message is available in the Receive Buffer and the Receive Interrupt Enable bit is HIGH (enabled).
	LOW (reset)	Receive Interrupt bit is automatically reset by a read access of Interrupt Register by the microcontroller.
Transmit Interrupt	HIGH (set)	This bit is set on a change of the Transmit Buffer Access from LOW to HIGH (released) and Transmit Interrupt Enable is HIGH (enabled).
	LOW (reset)	Transmit Interrupt bit will be reset after a read access of the Interrupt Register by the microcontroller.
Error Interrupt	HIGH (set)	This bit is set on a change of either the Error Status or Bus Status bits (see section 7.2.3) if the Error Interrupt Enable is HIGH (enabled).
	LOW (reset)	the Error Interrupt bit is reset by a read access of the Interrupt Register by the microcontroller.
Wake-Up Interrupt	HIGH (set)	The Wake-Up Interrupt bit is set HIGH, when the sleep mode is left (see 7.2.2).
	LOW (reset)	Wake-Up Interrupt bit is reset by a read access of Interrupt Register by the microcontroller.
Overrun Interrupt (note 2)	HIGH (set)	This bit is set HIGH, if both Receive Buffers contain a message and the first byte of another message should be stored (passed acceptance), and the Overrun Interrupt Enable is HIGH (enabled).
	LOW (reset)	Overrun Interrupt bit is reset by a read access of Interrupt Register by the microcontroller.

Notes to Table 7.10

1. Receive Interrupt bit (if enabled) and Receive Buffer Status bit (see section 7.2.3) are set at the same time.
2. Overrun Interrupt bit (if enabled) and Data Overrun bit (see section 7.2.3) are set at the same time.

7.2.5 Acceptance Code Register (ACR)

The Acceptance Code Register is part of the acceptance filter of the PCA82C200. This register can be accessed (read/write), if the Reset Request bit is set HIGH (present). When a message is received which passes the acceptance test and if there is an empty Receive Buffer, then the respective Descriptor and Data Field (see Fig 7.1) are sequentially stored in this empty buffer. In the case that there is no empty Receive Buffer, the Data Overrun bit is set HIGH (overrun), see sections 7.2.3 and 7.2.4. When the complete message has been correctly received the following occurs:

- the Receive Buffer Status bit is set HIGH (full)
- if the Receive Interrupt Enable bit is set HIGH (enabled), the Receive Interrupt is set HIGH (set).

The Acceptance Code bits (AC.7 - AC.0) and the eight most significant bits of the message's Identifier (ID.10 - ID.3) must be equal to those bit positions which are marked relevant by the Acceptance Mask bits (AM.7 - AM.0). If the following equation is satisfied, acceptance is given:

[(ID.10 .. ID.3)EQUAL(AC.7 .. AC.0)]OR
(AM.7 .. AM.0) = 1111 1111_b

Table 7.11 Acceptance Code Register bits.

ACR : ADDRESS 4							
7	6	5	4	3	2	1	0
AC.7	AC.6	AC.5	AC.4	AC.3	AC.2	AC.1	AC.0

During transmission of a message which passes the acceptance test, the message is also written to its own Receive Buffer. If no Receive Buffer is available Data Overrun is signalled because it is not known at the start of a message whether the PCA82C200 will lose arbitration and so become a receiver of the message.

7.2.6 Acceptance Mask Register (AMR)

The Acceptance Mask Register is part of the acceptance filter of the PCA82C200. This register can be accessed (read/write) if the Reset Request bit is set HIGH (present). The Acceptance Mask Register qualifies which of the corresponding bits of the acceptance code are 'relevant' or 'do not care' for acceptance filtering.

Table 7.12 Acceptance Mask Register bits.

AMR : ADDRESS 5							
7	6	5	4	3	2	1	0
AM.7	AM.6	AM.5	AM.4	AM.3	AM.2	AM.1	AM.0

Table 7.13 Description of the Acceptance Mask Register bits.

ACCEPTANCE MASK BIT	VALUE	COMMENTS
AM.7 to AM.0	HIGH LOW	This bit position is 'don't care' for the acceptance of a message. This bit position is 'relevant' for acceptance filtering.

7.2.7 Bus Timing Register 0 (BTR 0)

The contents of Bus Timing Register 0 defines the values of the Baud Rate Prescaler (BRP) and the Synchronization Jump Width (SJW). This register can be accessed (read/write) if the Reset Request bit is set HIGH (present).

Table 7.14 Bus Timing Register 0 bits.

BTR 0 : ADDRESS 6							
7	6	5	4	3	2	1	0
SJW.1	SJW.0	BRP.5	BRP.4	BRP.3	BRP.2	BRP.1	BRP.0

Baud Rate Prescaler (BRP)

The period of the system clock t_{SCL} is programmable and determines the individual bit timing. The system clock is calculated using the following equation:

$t_{SCL} = 2t_{CLK}(32BRP.5 + 16BRP.4 + 8BRP.3 + 4BRP.2 + 2BRP.1 + BRP.0 + 1)$

t_{CLK} = time period of the PCA82C200 oscillator.

Synchronization Jump Width (SJW)

To compensate for phase shifts between clock oscillators of different bus controllers, any bus controller must resynchronize on any relevant signal edge of the current transmission. The synchronization jump width defines the maximum number of clock cycles a bit period may be shortened or lengthened by one resynchronization:

$t_{SJW} = t_{SCL}(2SJW.1 + SJW.0 + 1)$

For further information on bus timing see sections 7.2.8 and 8.0.

7.2.8 Bus Timing Register 1 (BTR 1)

The contents of Bus Timing Register 1 defines the length of the bit period, the location of the sample point and the number of samples to be taken at each sample point. This register may be accessed (read/write) if the Reset Request bit is set HIGH (present).

Sampling (SAM)

Table 7.16 Selection of sampling

BIT	VALUE	COMMENTS
SAM	HIGH	Three samples are taken.
	LOW	The bus is sampled once.

SAM = logic 0 is recommended for high speed buses (SAE class C), while SAM = logic 1 is recommended for slow/medium speed buses (class A and B) where filtering of spikes on the bus-line is beneficial (see section 8.1.6).

Time Segment 1 (TSEG1) and Time Segment 2 (TSEG2)

TSEG1 and TSEG2 determine the number of clock cycles per bit period and the location of the sample point:

$t_{TSEG1} = t_{SCL}(8TSEG1.3 + 4TSEG1.2 + 2TSEG1.1 + TSEG1.0 + 1)$

$t_{TSEG2} = t_{SCL}(4TSEG2.2 + 2TSEG2.1 + TSEG2.0 + 1)$

For further information on bus timing see sections 7.2.7 and 8.0.

Table 7.15 Bus Timing Register 1 bits.

BTR 1 : ADDRESS 7							
7	6	5	4	3	2	1	0
SAM	TSEG2.2	TSEG2.1	TSEG2.0	TSEG1.3	TSEG1.2	TSEG1.1	TSEG1.0

7.2.9 Output Control Register (OCR)

The Output Control Register allows, under software control, the set-up of different output driver configurations. This register may be accessed (read/write) if the Reset Request bit is set HIGH (present).

If the PCA82C200 is in the sleep mode (Goto Sleep = HIGH) a recessive level is output on the TX0 and TX1 pins. If the PCA82C200 is in the reset state (Reset Request = HIGH) the output drivers are floating.

Table 7.17 Output Control Register bits.

OCR : ADDRESS 8							
7	6	5	4	3	2	1	0
OCTP1	OCTN1	OCPOL1	OCTP0	OCTN0	OCPOL0	OCMODE1	OCMODE0

Normal Output Mode

In Normal Output Mode the bit sequence (TXD) is sent via TX0 and TX1. The voltage levels on the output driver pins TX1 and TX0 depend on both the driver characteristic programmed by OCTPx, OCTNx (float, pull-up, pull-down, push-pull) and the output polarity programmed by OCPOLX.

Clock Output Mode

For the TX0 pin this is the same as in Normal Output Mode. However, the data stream to TX1 is replaced by the transmit clock (TXCLK). The rising edge of the transmit clock (non inverted) marks the beginning of a bit period. The clock pulse width is t_{SCL}.

Bi-phase Output Mode

In contrast to Normal Output Mode the bit representation is time variant and toggled. If the bus controllers are galvanically decoupled from the bus-line by a transformer, the bit stream is not allowed to contain a DC component. This is achieved by the following scheme. During recessive bits all outputs are deactivated (3-state). Dominant bits are sent alternatingly on TX0 and TX1, i.e. the first dominant bit is sent on TX0, the second is sent on TX1, and the third one is sent on TX0 again, etc.

Test Output Mode

For the TX0 pin this is the same as in Normal Output Mode. To measure the delay time of the transmitter and receiver this mode connects the output of the input comparator (COMP OUT) with the input of the output driver TX1. This mode is used for production testing only.

The following two tables, Table 7.18 and Table 7.19, show the relationship between the bits of the Output Control Register and the two serial output pins TX0 and TX1 of the PCA82C200, connected to the serial bus (see Fig.2.1).

Table 7.18 Description of the Output Mode bits.

OCMODE1	OCMODE0	DESCRIPTION
1	0	Normal Output Mode; TX0, TX1: bit sequence (TXD; note 1).
1	1	Clock Output Mode; TX0: bit sequence, TX1: bus clock (TXCLK).
0	0	Bi-phase Output Mode
0	1	Test Output Mode; TX0: bit sequence, TX1: COMP OUT

Note to Table 7.18

1. TXD is the data bit to be transmitted.

Table 7.19 Output pin set-up.

DRIVE	OCTPx	OCTNx	OCPOLx	TXD	TPx	TNx	TXx
Float	0	0	0	0	Off	Off	float
	0	0	0	1	Off	Off	float
	0	0	1	0	Off	Off	float
	0	0	1	1	Off	Off	float
Pull-down	0	1	0	0	Off	On	LOW
	0	1	0	1	Off	Off	float
	0	1	1	0	Off	Off	float
	0	1	1	1	Off	On	LOW
Pull-up	1	0	0	0	Off	Off	float
	1	0	0	1	On	Off	HIGH
	1	0	1	0	On	Off	HIGH
	1	0	1	1	Off	Off	float
Push/Pull	1	1	0	0	Off	On	LOW
	1	1	0	1	On	Off	HIGH
	1	1	1	0	On	Off	HIGH
	1	1	1	1	Off	On	LOW

Notes to Table 7.19

1. TPx is the on-chip output transistor x, connected to V_{DD}; x = 0 or 1.
2. TNx is the on-chip output transistor x, connected to V_{SS}; x = 0 or 1.
3. TXx is the serial output level on pin TX0 or TX1. It is required that the output level on the CAN bus is dominant with TXD = 0 and recessive with TXD = 1 (see section 9.1.1)

7.2.10 Test Register (TR)

The Test Register is used for production testing only.

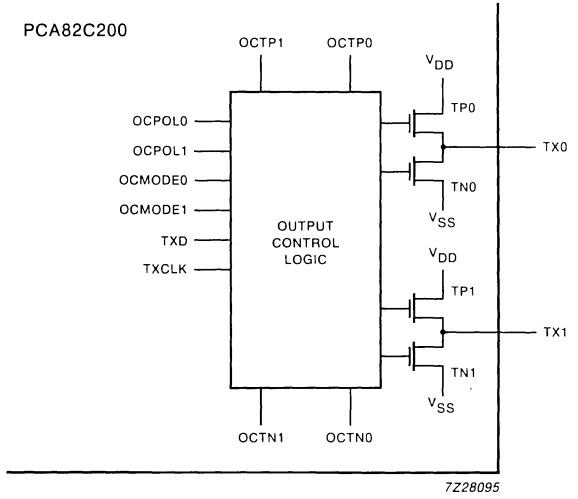


Fig.7.2 Configurable CAN transmitter.

7.3 Transmit Buffer layout

The global layout of the Transmit Buffer is shown in Fig.7.1. This buffer serves to store a message from the microcontroller to be transmitted by the PCA82C200. It is subdivided into Descriptor and Data Field. The Transmit Buffer can be written to and read from by the microcontroller (see note 1 to Table 7.3).

7.3.1 Descriptor

Table 7.20 Descriptor Byte 1 (DSCR1).

DSCR1 : ADDRESS 10							
7	6	5	4	3	2	1	0
ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3

Table 7.21 Descriptor Byte 2 (DSCR2).

DSCR2 : ADDRESS 11							
7	6	5	4	3	2	1	0
ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0

Identifier (ID)

The Identifier consists of 11 bits (ID.10 to ID.0). ID.10 is the most significant bit, which is transmitted first on the bus during the arbitration process. The Identifier acts as the message's name, used in a receiver for acceptance filtering, and also determines the bus access priority during the arbitration process. The lower the binary value of the Identifier the higher the priority. This is due to the larger number of leading dominant bits during arbitration (see section 5).

Remote Transmission Request bit (RTR)

Table 7.22 Description of the RTR bit.

BIT	VALUE	COMMENTS
RTR	HIGH	Remote Frame will be transmitted by the PCA82C200
	LOW	Data Frame will be transmitted by the PCA82C200.

Data Length Code (DLC)

The number of bytes (Data Byte Count) in the Data Field of a message is coded by the Data Length Code. At the start of a Remote Frame transmission the Data Length Code is not considered due to the RTR bit being HIGH (remote). This forces the number of transmitted/received data bytes to be 0. Nevertheless, the Data Length Code must be specified correctly to avoid bus errors, if two CAN-controllers start a Remote Frame transmission simultaneously.

The range of the Data Byte Count is 0 to 8 bytes and coded as follows:

Data Byte Count = 8DLC.3 + 4DLC.2 + 2DLC.1 + DLC.0

For reasons of compatibility no Data Byte Counts other than 0 to 8 should be used.

7.3.2 Data Field

The number of transferred data bytes is determined by the Data Length Code. The first bit transmitted is the most significant bit of data byte 1 at address 12.

7.4 Receive Buffer layout

The layout of the Receive Buffer and the individual bytes correspond to the definitions given for the Transmit Buffer layout, except that the addresses start at 20 instead of 10 (see Fig 7.1).

7.5 Clock Divider Register (CDR)

The Clock Divider Register controls the CLK OUT frequency for the microcontroller (see Fig.2.1). It can be written to or read by the microcontroller. The default state of the register is divided by 12 for Motorola mode and divided by 2 for Intel mode. Values from 0 to 7 may be written into this register and will result in the CLK OUT frequencies shown in Table 7.24.

Table 7.23 Clock Divider Register bits.

CDR : ADDRESS 31							
7	6	5	4	3	2	1	0
-	-	-	-	-	CD.2	CD.1	CD.0

Bits CDR.7 to CDR.3 are reserved.

Table 7.24 CLK OUT frequency selection.

CD.2	CD.1	CD.0	CLK OUT FREQUENCY
0	0	0	$f_{CLK}/2$
0	0	1	$f_{CLK}/4$
0	1	0	$f_{CLK}/6$
0	1	1	$f_{CLK}/8$
1	0	0	$f_{CLK}/10$
1	0	1	$f_{CLK}/12$
1	1	0	$f_{CLK}/14$
1	1	1	f_{CLK}

Note: f_{CLK} is the frequency of the oscillator

8.0 BUS TIMING/SYNCHRONIZATION

The Bus Timing Logic (BTL) monitors the serial bus-line via the on-chip input comparator and performs the following functions (see section 2):

- monitors the serial bus-line level
- adjusts the sample point, within a bit period (programmable)
- samples the bus-line level using majority logic (programmable, 1 or 3 samples)
- Synchronization to the bit stream:
  - Hard synchronization at the start of a message
  - Resynchronization during transfer of a message.

The configuration of the BTL is performed during the initialization of the PCA82C200. The BTL uses the following three registers:

- Control register (Synch)
- Bus Timing Register 0
- Bus Timing Register 1.

8.1 Bit timing

A bit period is built up from a number of system clock cycles (t_{SCL}), see section 7.2.7. One bit period is the result of the addition of the programmable segments TSEG1 and TSEG2 and the general segment SYNCSEG (see sections 7.2.7 to 7.2.8).

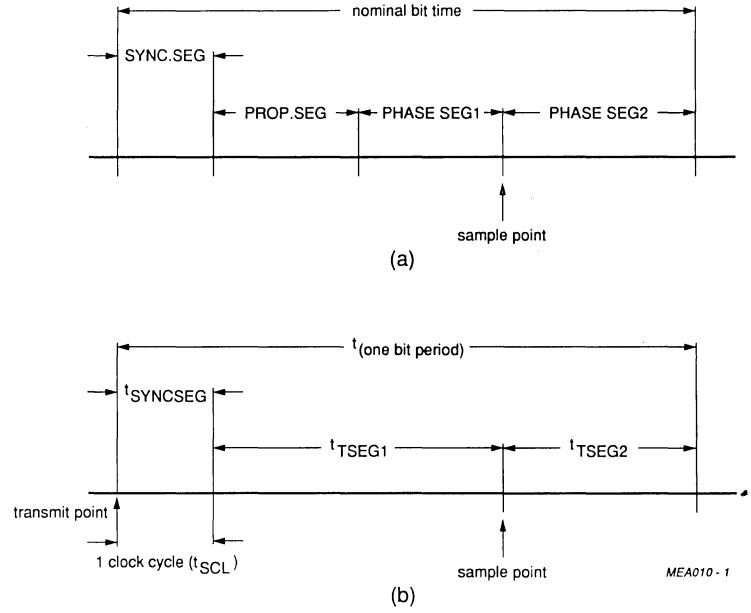


Fig.8.1 (a) Bit period as defined by the CAN protocol.
(b) Bit period as implemented in the PCA82C200.

8.1.1 SYNCHRONIZATION SEGMENT (SYNCSEG)

The incoming edge of a bit is expected during this state; this state corresponds to one system clock cycle ($1 \times t_{SCL}$).

8.1.2 TIME SEGMENT 1 (TSEG1)

This segment determines the location of the sampling point within a bit period, which is at the end of TSEG1. TSEG1 is programmable from 1 to 16 system clock cycles (see section 7.2.8).

The correct location of the sample point is essential for the correct functioning of a transmission. The following points must be taken into consideration:

- A Start-Of-Frame (see section 9.2.1) causes all PCA82C200's to perform a 'hard synchronization' (see section 8.2.1) on the first recessive-to-dominant edge. During arbitration, however, several PCA82C200's may simultaneously transmit. Therefore it may require twice the sum of bus-line, input comparator and the output driver delay times until the bus is stable. This is the propagation delay.
- To avoid sampling at an incorrect position, it is necessary to include an additional synchronization buffer on both sides of the sample point. The main reasons for incorrect sampling are:
  - incorrect synchronization due to spikes on the bus-line
  - slight variations in the oscillator frequency of each PCA82C200 in the network, which results in a phase error.

Time Segment 1 consists of the segment for compensation of propagation delays and the synchronization buffer directly before the sample point (see Fig.8.1).

8.1.3 TIME SEGMENT 2 (TSEG2)

This time segment provides:

- additional time at the sample point for calculation of the subsequent bit levels (e.g. arbitration)
- synchronization buffer segment directly after the sample point (see section 8.1.2).

TSEG2 is programmable from 1 to 8 system clock cycles (see section 7.2.8).

8.1.4 SYNCHRONIZATION JUMP WIDTH (SJW)

SJW defines the maximum number of clock cycles (t_{SCL}) a bit period may be reduced or increased by one resynchronization. SJW is programmable from 1 to 4 system clock cycles (see section 7.2.7).

8.1.5 PROPAGATION DELAY TIME (t_{prop}).

The propagation delay time is calculated by summing the maximum propagation delay times of the physical bus, the input comparator and the output driver. The resulting sum is multiplied by 2 and then rounded up to the nearest multiple of t_{SCL}.

tprop = 2 x (physical bus delay + input comparator delay + output driver delay)

8.1.6 BIT TIMING RESTRICTIONS

Restrictions on the configuration of the bit timing are based on internal processing. The restrictions are:

- $t_{TSEG2} \geq 2t_{SCL}$
- $t_{TSEG2} \geq t_{SJW}$
- $t_{TSEG1} \geq t_{TSEG2}$
- $t_{TSEG1} \geq t_{SJW} + t_{prop}$

The three sample mode (SAM = 1) has the effect of introducing a delay of one system clock cycle on the bus-line. This must be taken into account for the correct calculation of TSEG1 and TSEG2:

- $t_{TSEG1} \geq t_{SJW} + t_{prop} + 2t_{SCL}$
- $t_{TSEG2} \geq 3t_{SCL}$

8.2 Synchronization

Synchronization is performed by a state machine which compares the incoming edge with its actual bit timing and adapts the bit timing by hard synchronization or resynchronization.

8.2.1 HARD SYNCHRONIZATION

This type of synchronization occurs only at the beginning of a message. The PCA82C200 synchronizes on the first incoming recessive-to-dominant edge of a message (being the leading edge of a message's Start-Of-Frame bit; see section 8.1).

8.2.2 RESYNCHRONIZATION

Resynchronization occurs during the transmission of a message's bit stream to compensate for:

- variations in individual PCA82C200 oscillator frequencies
- changes introduced by switching from one transmitter to another (e.g. during arbitration).

As a result of resynchronization either t_{TSEG1} may be increased by up to a maximum of t_{SJW} or t_{TSEG2} may be decreased by up to a maximum of t_{SJW}:

- $t_{TSEG1} \leq t_{SCL}((TSEG1 + 1) + (SJW + 1))$
- $t_{TSEG2} \geq t_{SCL}((TSEG2 + 1) - (SJW + 1))$

Note: TSEG1, TSEG2 and SJW are the programmed numerical values.

The phase error (e) of an edge is given by the position of the edge relative to SYNCSEG, measured in system clock cycles (t_{SCL}). The value of the phase error is defined as:

- e = 0, if the edge occurs within SYNCSEG
- e > 0, if the edge occurs within TSEG1
- e < 0, if the edge occurs within TSEG2.

The effect of resynchronization is:

- the same as that of a hard synchronization, if the magnitude of the phase error (e) is less or equal to the programmed value of t_{SJW} (see section 7.2.7)
- to increase a bit period by the amount of t_{SJW}, if the phase error is positive and the magnitude of the phase error is larger than t_{SJW}.
- to decrease a bit period by the amount of t_{SJW}, if the phase error is negative and the magnitude of the phase error is larger than t_{SJW}.

8.2.3 SYNCHRONIZATION RULES

The synchronization rules are as follows:

- only one synchronization within one bit time is used.
- an edge is used for synchronization only if the value detected at the previous sample point differs from the bus value immediately after the edge
- hard synchronization is performed whenever there is a recessive-to-dominant edge during Bus-Idle (see section 8)
- all other edges (recessive-to-dominant and optionally dominant-to-recessive edges if the Synch bit is set HIGH; see section 7.2.1) which are candidates for resynchronization will be used with the following exception:

  - A transmitting PCA82C200 will not perform a resynchronization as a result of a recessive-to-dominant edge with positive phase error, if only these edges are used for resynchronization. This ensures that the delay times of the output driver and input comparator do not cause a permanent increase in the bit time.

9.0 COMMUNICATION PROTOCOL

9.1 Frame types

The PCA82C200 bus controller supports the four different CAN protocol frame types for communication:

- Data Frame, to transfer data
- Remote Frame, request for data
- Error Frame, globally signal a (locally) detected error condition
- Overload Frame, to extend delay time of subsequent frames (an Overload Frame is not initiated by the PCA82C200).

9.1.1 BIT REPRESENTATION

There are two logical bit representations used in the CAN protocol:

- A recessive bit on the bus-line appears only if all connected PCA82C200's send a recessive bit at that moment
- Dominant bits always overwrite recessive bits i.e. the resulting bit level on the bus-line is dominant.

9.2. Data Frame

A Data Frame carries data from a transmitting PCA82C200 to one or more receiving PCA82C200's. A Data Frame is composed of seven different bit-fields:

- Start-Of-Frame
- Arbitration Field
- Control Field
- Data Field (may have a length of zero)
- CRC Field
- Acknowledge Field
- End-Of-Frame.

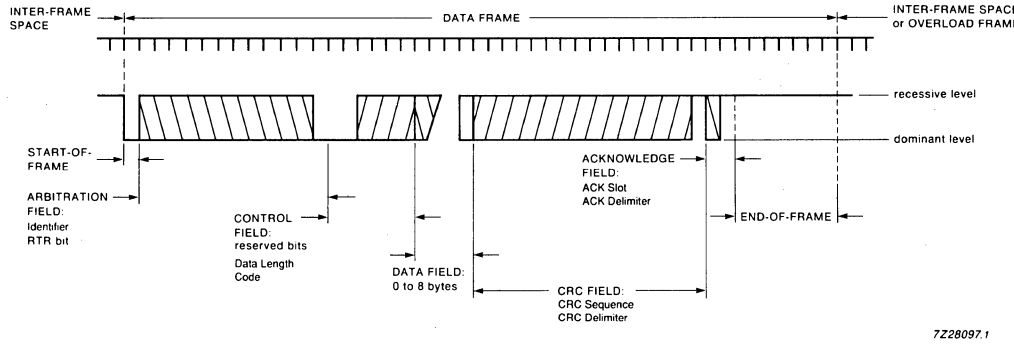


Fig.9.1 Data Frame.

9.2.1 Start-Of-Frame bit

Signals the start of a Data Frame or Remote Frame. It consists of a single dominant bit used for hard synchronization of a PCA82C200 in receive mode.

9.2.2 Arbitration Field

Consists of the message Identifier and the RTR bit (see section 7.3.1). In the event of simultaneous message transmissions by two or more PCA82C200's the bus access conflict is solved by bit-wise arbitration, which is active during the transmission of the Arbitration Field.

Identifier

This 11-bit field is used to provide information about the message, as well as the bus access priority. It is transmitted in the order ID.10 to ID.0 (LSB). The situation that the seven most significant bits (ID.10 to ID.4) are all recessive must not occur.

An Identifier does not define which particular PCA82C200 will receive the frame, because a CAN base communication network does not discriminate between a point-to-point, multicast or broadcast communication.

Remote Transmission Request bit (RTR)

A PCA82C200, acting as a receiver for certain information may initiate the transmission of the respective data by transmitting a Remote Frame to the network, addressing the data source via the Identifier and setting the RTR bit HIGH (remote; recessive bus level). If the data source simultaneously transmits a Data Frame containing the requested data, it uses the same Identifier. No bus access conflict occurs due to the RTR bit being set LOW (data; dominant bus level) in the Data Frame.

9.2.3 Control Field

This field consists of six bits. It includes two reserved bits (for future expansions of the CAN-protocol), transmitted with a dominant bus level, and is followed by the Data Length Code (4 bits). The number of bytes in the (destuffed; number of data bytes to be transmitted/ received) Data Field is indicated by the Data Length Code. Admissible values of the Data Length Code and hence the number of bytes in the (destuffed) Data Field, are 0 to 8. A logic 0 (logic 1) in the Data Length Code is transmitted as a dominant (recessive) bus level, respectively.

9.2.4 Data Field

The data, stored within the Data Field of the Transmit Buffer, are transmitted according to the Data Length Code. Conversely, data of a received Data Frame will be stored in the Data Field of a Receive Buffer. Data is stored byte-wise both for transmission by the microcontroller and on reception by the PCA82C200. The most significant bit of the first data byte (lowest address) is transmitted/received first.

9.2.5 Cyclic Redundancy Code Field (CRC)

The CRC Field consists of the CRC Sequence (15 bits) and the CRC Delimiter (1 recessive bit). The Cyclic Redundancy Code (CRC) encloses the destuffed bit stream of the Start-Of-Frame, Arbitration Field, Control Field , Data Field and CRC Sequence. The most significant bit of the CRC Sequence is transmitted/ received first. This frame check sequence, implemented in the PCA82C200, is derived from a cyclic redundancy code best suited for frames with a total bit count of less than 127 bits, see section 9.8.3. With Start-Of-Frame (dominant bit) included in the code word, any rotation of the code word can be detected by the absence of the CRC Delimiter (recessive bit).

9.2.6 Acknowledge Field (ACK)

The Acknowledge Field consists of two bits, the Acknowledge Slot and the Acknowledge Delimiter, which are transmitted with a recessive level by the transmitter of the Data Frame. All PCA82C200's having received the matching CRC Sequence, report this by overwriting the transmitter's recessive bit in the Acknowledge Slot with a dominant bit (see section 9.9.2). Thereby a transmitter, still monitoring the bus level recognizes that at least one receiver within the network has received a complete and correct message (i.e. no error was found). The Acknowledge Delimiter (recessive bit) is the second bit of the Acknowledge Field. As a result, the Acknowledge Slot is surrounded by two recessive bits: the CRC Delimiter and the Acknowledge Delimiter.

All nodes within a CAN network may use all the information coming to the network by the PCA82C200's (shared memory concept). Therefore, acknowledgement and error handling are defined to provide all information in a consistent way throughout this shared memory. Hence, there is no reason to discriminate different receivers of a message in the acknowledge field. If a node is disconnected from the network due to bus failure, this particular node is no longer part of the shared memory. To identify a "lost node" additional and application specific precautions are required.

9.2.7 End-Of-Frame

Each Data Frame or Remote Frame is delimited by the End-Of-Frame bit sequence which consists of seven recessive bits (exceeds the bit stuff width by two bits). Using this method a receiver detects the end of a frame independent of a previous transmission error because the receiver expects all bits up to the end of the CRC sequence to be coded by the method of bit-stuffing (see section 9.7.3 Coding/Decoding). The bit-stuffing logic is deactivated during the End-Of-Frame sequence.

9.3 Remote Frame

A PCA82C200, acting as a receiver for certain information may initiate the transmission of the respective data by transmitting a Remote Frame to the network, addressing the data source via the Identifier and setting the RTR bit HIGH (remote; recessive bus level). The Remote Frame is similar to the Data Frame with the following exceptions:

- RTR bit is set HIGH
- Data Length Code is ignored.
- no Data Field contained.

Note that the Data Length Code value should be the same as for the corresponding Data Frame (although this is ignored for a Remote Frame).

A Remote Frame is composed of six different bit fields:

- Start-Of-Frame
- Arbitration Field
- Control Field
- CRC-Field
- Acknowledge Field
- End-Of-Frame.

See section 9.2 for a more detailed explanation of the Remote Frame bit fields.

9.4 Error Frame

The Error Frame consists of two different fields. The first field is accomplished by the superimposing of Error Flags contributed from different PCA82C200s. The second field is the Error Delimiter (see Fig.9.2).

9.4.1 Error Flag

There are two forms of an Error Flag:

- Active Error Flag, consists of six consecutive dominant bits
- Passive Error Flag, consists of six consecutive recessive bits unless it is overwritten by dominant bits from other PCA82C200's.

An error-active PCA82C200 (see section 9.9) detecting an error condition signals this by transmission of an Active Error Flag. This Error Flag's form violates the bit-stuffing law (see section 9.7.3) applied to all fields, from Start-Of-Frame to CRC Delimiter, or destroys the fixed form of the fields Acknowledge Field or End-Of-Frame (see Fig 9.1). Consequently, all other PCA82C200's detect an error condition and start transmission of an Error Flag. Therefore the sequence of dominant bits, which can be monitored on the bus, results from a superposition of different Error Flags transmitted by individual PCA82C200's. The total length of this sequence varies between six (min) and twelve (max) bits.

An error-passive PCA82C200 (see section 9.9) detecting an error condition tries to signal this by transmission of a Passive Error Flag. The error-passive PCA82C200 waits for six consecutive bits with identical polarity, beginning at the start of the Passive Error Flag. The Passive Error Flag is complete when these six identical bits have been detected.

9.4.2 Error Delimiter

The Error Delimiter consists of eight recessive bits and has the same format as the Overload Delimiter. After transmission of an Error Flag, each PCA82C200 monitors the bus-line until it detects a transition from a dominant-to-recessive bit level. At this point in time, every PCA82C200 has finished sending its Error Flag and all PCA82C200's start transmission of seven recessive bits (plus the recessive bit at dominant-to-recessive transition, results in a total of eight recessive bits). After this event and an Intermission Field all error-active PCA82C200's within the network can start a transmission simultaneously.

If a detected error is signalled during transmission of a Data Frame or Remote Frame, the current message is spoiled and a retransmission of the message is initiated.

If a PCA82C200 monitors any deviation of the Error Frame, a new Error Frame will be transmitted. Several consecutive Error Frame's may result in the PCA82C200 becoming error-passive and leaving the network unblocked.

In order to terminate an Error Flag correctly, an error-passive CAN bus controller requires the bus to be Bus-Idle (see section 9.6.2) for at least three bit periods (if there is a local error at an error-passive receiver). Therefore a CAN bus should not be 100% permanently loaded.

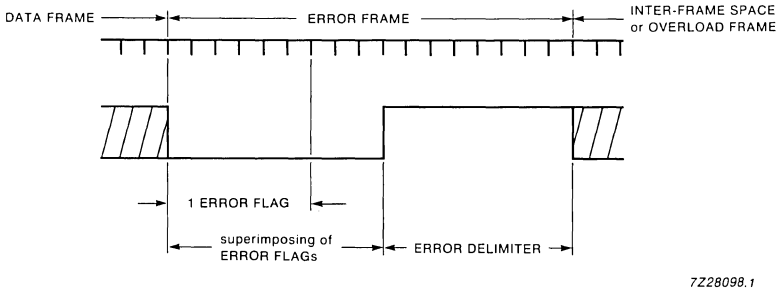


Fig.9.2 Error Frame.

9.5 Overload Frame

The Overload Frame consists of two fields, the Overload Flag and the Overload Delimiter. There are two conditions in the CAN-protocol which lead to the transmission of an Overload Flag:

- condition 1; receiver circuitry require more time to process the current data before receiving the next frame (receiver not ready)
- condition 2; detection of a dominant bit during Intermission Field (see section 9.6.1).

The transmission of an Overload Frame may only start:

- condition 1; during the first bit period of an expected Intermission Field
- condition 2; one bit period after detecting the dominant bit during Intermission Field.

The PCA82C200 will never initiate transmission of a condition 1 Overload Frame and will only react on a transmitted condition 2 Overload Frame, according to the CAN protocol. No more than two Overload Frames are generated to delay a Data Frame or a Remote Frame. Although the overall form of the Overload Frame corresponds to that of the Error Frame, an Overload Frame does not initiate or require the retransmission of the preceding frame.

9.5.1 Overload Flag

The Overload Flag consists of six dominant bits and has a similar format to the Error Flag.

The Overload Flag's form corrupts the fixed form of the Intermission Field. All other PCA82C200's detecting the overload condition also transmit an Overload Flag (condition 2).

9.5.2 Overload Delimiter

The Overload Delimiter consists of eight recessive bits and takes the same form as the Error Delimiter. After transmission of an Overload Flag, each PCA82C200 monitors the bus-line until it detects a transition from a dominant-to-recessive bit level. At this point in time, every PCA82C200 has finished sending its Overload Flag and all PCA82C200's start simultaneously transmitting seven more recessive bits.

9.6 Inter-Frame Space

Data Frames and Remote Frames are separated from preceding frames (all types) by an Inter-Frame Space, consisting of an Intermission Field and a Bus-Idle. Error-passive PCA82C200's also send a Suspend Transmission (see 9.9.5) after transmission of a message. Overload Frames and Error Frames are not preceded by an Inter-Frame Space.

9.6.1 Intermission Field

The Intermission Field consists of three recessive bits. During an Intermission period, no frame transmissions will be started by any PCA82C200. An Intermission is required to have a fixed time period to allow a CAN-controller to execute internal processes prior to the next receive or transmit task.

9.6.2 Bus-Idle

The Bus-Idle time may be of arbitrary length (min. 0 bit). The bus is recognized to be free and a CAN-controller having information to transmit may access the bus. The detection of a dominant bit level during Bus-Idle on the bus is interpreted as the Start-Of-Frame.

9.7 Bus organization

Bus organization is based on five basic rules described in the following paragraphs.

9.7.1 Bus access

PCA82C200's only start transmission during the Bus-Idle state. All PCA82C200's synchronize on the leading edge of the Start-Of-Frame (hard synchronization).

9.7.2 Arbitration

If two or more PCA82C200's simultaneously start transmitting, the bus access conflict is solved by a bit-wise arbitration process during transmission of the Arbitration Field.

During arbitration every transmitting PCA82C200 compares its transmitted bit level with the monitored bus level. Any PCA82C200 which transmits a recessive bit and monitors a dominant bus level immediately becomes the receiver of the higher priority message on the bus without corrupting any information on the bus.
Each message contains an unique Identifier and a RTR bit describing the type of data within the message. The Identifier together with the RTR bit implicitly define the message's bus access priority. During arbitration the most significant bit of the Identifier is transmitted first and the RTR bit last. The message with the lowest binary value of the Identifier and RTR bit has the highest priority.

A Data Frame has higher priority than a Remote Frame due to its RTR bit having a dominant level.

For every Data Frame there is an unique transmitter. For reasons of compatibility with other CAN-bus controllers, use of the Identifier bit pattern ID = $1111111XXXX_b$ (X being bits of arbitrary level) is forbidden. The number of available different Identifiers is 2032 ($2^{11} - 2^4$).

9.7.3 Coding/decoding

The following bit fields are coded using the bit-stuffing technique:

- Start-Of-Frame
- Arbitration Field
- Control Field
- Data Field
- CRC Sequence.

When a transmitting PCA82C200 detects five consecutive bits of identical polarity to be transmitted, a complementary (stuff) bit is inserted into the transmitted bit-stream.

When a receiving PCA82C200 has monitored five consecutive bits with identical polarity in the received bit streams of the above described bit fields, it automatically deletes the next received (stuff) bit. The level of the deleted stuff bit has to be the complement of the previous bits; otherwise a Stuff Error will be detected and signalled (see section 9.8.2).

The remaining bit fields or frames are of fixed form and are not coded or decoded by the method of bit-stuffing.

The bit-stream in a message is coded according to the Non-Return-to-Zero (NRZ) method, i.e. during a bit period, the bit level is held constant, either recessive or dominant.

9.7.4 Error signalling

A PCA82C200 which detects an error condition, transmits an Error Flag. Whenever a Bit Error, Stuff Error, Form Error or an Acknowledgement Error is detected, transmission of an Error Flag is started at the next bit.

Whenever a CRC Error is detected, transmission of an Error Flag starts at the bit following the Acknowledge Delimiter, unless an Error Flag for another error condition has already started. An Error Flag violates the bit-stuffing law or corrupts the fixed form bit fields. A violation of the bit-stuffing law affects any PCA82C200 which detects the error condition. These devices will also transmit an Error Flag.

An error-passive PCA82C200 (see section 9.9) which detects an error condition, transmits a Passive Error Flag. A Passive Error Flag is not able to interrupt a current message at different PCA82C200's, but this type of Error Flag may be ignored by other PCA82C200's. After having detected an error condition, an error-passive PCA82C200 will wait for six consecutive bits with identical polarity and when monitoring them, interpret them as an Error Flag.

After transmission of an Error Flag, each PCA82C200 monitors the bus-line until it detects a transition from a dominant-to-recessive bit level. At this point in time, every PCA82C200 has finished transmitting its Error Flag and all PCA82C200's start transmitting seven additional recessive bits (Error Delimiter, see section 9.4.2).

The message format of a Data Frame or Remote Frame is defined in such a way, that all detectable errors can be signalled within the message transmission time and therefore, it is very simple for a PCA82C200 to associate an Error Frame to the corresponding message and to initiate retransmission of the corrupted message.

If a PCA82C200 monitors any deviation of the fixed form of an Error Frame, it transmits a new Error Frame.

9.7.5 Overload signalling

Some CAN-controllers (but not the PCA82C200) require to delay the transmission of the next Data Frame or Remote Frame by transmitting one or more Overload Frames. The transmission of an Overload Frame must start during the first bit of an expected Intermission. Transmission of Overload Frames which are reactions on a dominant bit during an expected Intermission Field, start one bit after this event.

Though the format of Overload Frame and Error Frame are identical, they are treated differently. Transmission of an Overload Frame during Intermission Field does not initiate the retransmission of any previous Data Frame or Remote Frame.

If a CAN-controller which transmitted an Overload Frame monitors any deviation of its fixed form, it transmits an Error Frame.

9.8 Error detection

The processes described in the following paragraphs are implemented in the PCA82C200 for error detection.

9.8.1 Bit Error

A transmitting PCA82C200 monitors the bus on a bit-by-bit basis. If the bit level monitored is different from the transmitted one, a Bit Error is signalled. The exceptions being:

- During the Arbitration Field, a recessive bit can be overwritten by a dominant bit. In this case, the PCA82C200 interprets this as a loss of arbitration
- During the Acknowledge Slot, only the receiving PCA82C200's are able to recognize a Bit Error.

9.8.2 Stuff Error

The following bit fields are coded using the bit-stuffing technique:

- Start-Of-Frame
- Arbitration Field
- Control Field
- Data Field
- CRC Sequence.

There are two possible ways of generating a Stuff Error:

- The disturbance generates more than the allowed five consecutive bits with identical polarity. These errors are detected by all PCA82C200's.
- A disturbance falsifies one or more of the five bits preceding the stuff bit. This error situation is not recognized as a Stuff Error by the receivers. Therefore, other error detection processes may detect this error condition such as: CRC check, format violation at the receiving PCA82C200's or Bit Error detection by the transmitting PCA82C200.

9.8.3 CRC Error

To ensure the validity of a transmitted message all receivers perform a CRC check. Therefore, in addition to the (destuffed) information digits (Start-Of-Frame up to Data Field), every message includes some control digits (CRC Sequence; generated by the transmitting PCA82C200 of the respective message) used for error detection.

The code used for the PCA82C200 bus controller is a (shortened) BCH code, extended by a parity check and has the following attributes:

- 127 bits as maximum length of the code
- 112 bits as maximum number of information digits (max. 83 bits are used by PCA82C200)
- length of the CRC Sequence amounts to 15 bits
- Hamming distance d = 6.

As a result, (d-1) random errors are detectable (some exceptions exist).

The CRC Sequence is calculated by the following procedure:

1. The destuffed bit stream consisting of Start-of-Frame up to the Data Field (if present) is interpreted as a polynomial with coefficients of 0 or 1.
2. This polynomial is divided (modulo-2) by the following generator polynomial:

$$f(X) = (X^{14} + X^9 + X^8 + X^6 + X^5 + X^4 + X^2 + X + 1)(X + 1) = 1100010110011001_b.$$

The remainder of this polynomial division is the CRC Sequence which includes a parity check. Burst errors are detected up to a length of 15 (degree of f(X)). Multiple errors (number of disturbed bits at least d = 6) are not detected with a residual error probability of 2^{-15} ($\simeq 3 \cdot 10^{-5}$) by CRC check only.

9.8.4 Form Error

Form Errors result from violation of the fixed form of the following bit fields:

- End-Of-Frame
- Intermission
- Acknowledge Delimiter
- CRC Delimiter.

During the transmission of these bit fields an error condition is recognized if a dominant bit level instead of a recessive one is detected.

9.8.5 Acknowledgement Error

This is detected by a transmitter whenever it does not monitor a dominant bit during the Acknowledge Slot.

9.8.6 Error detection by an Error Flag of another PCA82C200

The detection of an error is signalled by transmitting an Error Flag. An Active Error Flag causes a Stuff Error, a Bit Error or a Form Error at all other PCA82C200's.

9.8.7 Error detection capabilities

Errors which occur at all PCA82C200's (global errors) are 100% detected. For local errors, i.e. for errors occurring at some PCA82C200's only, the shortened BCH code, extended by a parity check, has the following error detection capabilities:

- Up to five single bit errors are 100% detected, even if they are distributed randomly within the code
- All single bit errors are detected if their total number (within the code) is odd
- The residual error probability of the CRC check amounts to 3×10^{-5}. As an error may be detected not only by CRC check but also by other detection processes described in sections 9.8.1 to 9.8.5, the residual probability is several magnitudes less than 3×10^{-5} for undetected errors.

9.9 Error confinement (definitions)

9.9.1 Bus-Off

A PCA82C200 which has too many unsuccessful transmissions, relative to the number of successful transmissions, will enter the Bus-Off state (see section 9.10.3). It remains in this state, neither receiving nor transmitting messages until the Reset Request bit is set LOW (absent) and both Error Counters are set to '0' (see note 5 to Table 7.8 and section 9.10.3).

9.9.2 ACKNOWLEDGE (ACK)

A PCA82C200 which has received a valid message correctly, indicates this to the transmitter by transmitting a dominant bit level on the bus during the Acknowledge Slot, independent of accepting or rejecting the message.

9.9.3 ERROR-ACTIVE

An error-active PCA82C200 is in its normal operating state able to receive and to transmit normally and also to transmit an Active Error Flag (see section 9.10.3).

9.9.4 ERROR-PASSIVE

An error-passive PCA82C200 may transmit or receive messages normally. In the case of a detected error condition it transmits a Passive Error Flag, instead of an Active Error Flag. Hence the influence on bus activities by an error-passive PCA82C200 (e.g. due to a malfunction) is reduced.

9.9.5 SUSPEND TRANSMISSION

After an error-passive PCA82C200 has transmitted a message, it sends eight recessive bits after the Intermission Field and then checks for Bus-Idle. If during Suspend Transmission another PCA82C200 starts transmitting a message the suspended PCA82C200 will become the receiver of this message; otherwise being in Bus-Idle it may start to transmit a further message.

9.9.6 START-UP

A PCA82C200 which either was switched off or is in the Bus-Off state, must run a start-up routine in order to:

- Synchronize with other available PCA82C200's, before starting to transmit. Synchronizing is achieved, when 11 recessive bits, equivalent to Acknowledge Delimiter, End-Of-Frame and Intermission Field, have been detected (Bus-Free).
- Wait for other PCA82C200's without passing into the Bus-Off state (due to a missing acknowledge), if there is no other PCA82C200 currently available.

9.10 Aims of error confinement

9.10.1 DISTINCTION OF SHORT AND LONG LASTING DISTURBANCES

The microcontroller must be informed when there are long-lasting disturbances and when bus activities have returned to normal operation. During long lasting disturbances, a PCA82C200 enters the Bus-Off state and the microcontroller may use default values.

Minor disturbances of bus activities will not affect a PCA82C200. In particular, a PCA82C200 does not enter the Bus-Off state or inform the microcontroller of a short lasting bus disturbance.

9.10.2 DETECTION AND LOCALIZATION OF HARDWARE DISTURBANCES AND DEFECTS

The rules for error confinement are defined by the CAN protocol specification (and implemented in the PCA82C200), in that the PCA82C200, being nearest to the error-locus, reacts with a high probability the quickest (i.e. becomes error-passive or Bus-Off). Hence errors can be localized and their influence on normal bus activities is minimized.

9.10.3 ERROR CONFINEMENT

All PCA82C200's contain a Transmit Error Counter and a Receive Error Counter, which registers errors during the transmission and the reception of messages, respectively.

If a message is transmitted or received correctly, the count is decreased. In the event of an error, the count is increased. The Error Counters have an non-proportional method of counting: an error causes a larger counter increase than a correctly transmitted/received message causes the count to decrease. Over a period of time this may result in an increase in error counts, even if there are fewer corrupted messages than uncorrupted ones. The level of the Error Counters reflect the relative frequency of disturbances. The ratio of increase/decrease depends on the acceptable ratio of invalid/valid messages on the bus and is hardware implemented to eight.

If one of the Error Counters exceeds the Warning Limit of 96 error points, indicating a significant accumulation of error conditions, this is signalled by the PCA82C200 (Error Status, Error Interrupt).

A PCA82C200 operates in the error-active mode until it exceeds 127 error points on one of its Error Counters. At this point it will enter the error-passive state.

A transmit error which exceeds 255 error points results in the PCA82C200 entering the Bus-Off state.

10.0 LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC134).

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	Supply voltage range	4.5	5.5	V
$\pm I_I$	Input/output current on any pin except from TX0 and TX1	–	10	mA
I_{OT}	Sink current of TX0 and TX1 together; see note 1	–	28	mA
$-I_{OT}$	Source current of TX0 and TX1 together	–	20	mA
T_{amb}	Operating ambient temperature range	–40	+125	°C
T_{stg}	Storage temperature range	–65	+150	°C
P_{tot}	Total power dissipation; see note 2	–	1	W

Notes

1. I_{OT} is allowed in case of a bus failure condition because then the TX-outputs are switched off automatically after a short time (Bus-Off state). During normal operation I_{OT} is a peak current, permitted for $t < 100$ ms. The average output current must not exceed 10 mA for each TX-output.
2. The value is based on the maximum allowable die temperature and the thermal resistance of the package, not on device power consumption.

11.0 DC CHARACTERISTICS

V_{DD} = 5 V ±10%; V_{SS} = 0 V; T_{amb} = –40 to +125 °C; unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{IL1}	Input voltage LOW	all inputs; except XTAL1, RX0 and RX1	–0.5	0.8	V
V_{IL2}	XTAL1 input voltage LOW		–	$0.2V_{DD}$	V
V_{IH1}	Input voltage HIGH	all inputs; except XTAL1, $\overline{RST}$, RX0 and RX1	3.2	V_{DD} + 0.5	V
V_{IH2}	XTAL1 input voltage HIGH		$0.7V_{DD}$	–	V
V_{IH3}	$\overline{RST}$ input voltage HIGH		$0.7V_{DD}$	V_{DD} + 0.5	V
V_{OL}	Output voltage LOW	I_{OL} = 1.6 mA; all outputs except XTAL2, TX0 and TX1	–	0.45	V
V_{OH1}	Output voltage HIGH	I_{OH} = –80 µA; all outputs except TX0, TX1, $\overline{INT}$ and CLK OUT	2.4	–	V
V_{OH2}	CLK OUT voltage HIGH	I_{OH} = –80 µA	$0.8V_{DD}$	–	V
$\pm I_{LI}$	Input leakage current	0.45 < V_I < V_{DD}; all inputs except XTAL1, RX0 and RX1	–	10	µA
I_{DD}	Supply current	f_{CLK} = 16 MHz; $\overline{RST}$ = R_{SS}; see note 1	–	15	mA
I_{sm}	Sleep mode supply current	oscillator inactive; see note 2	–	40	µA
CAN input comparator					
		V_{DD} = 5 V ±5%; 1.4 V < V_I < V_{DD} – 1.4 V			
$\pm V_{DIF}$	Differential input voltage	note 3	42	–	mV
V_{HYST}	Hysteresis voltage	note 3	12	45	mV
$\pm I_I$	Input current		–	400	nA
CAN output driver					
		V_{DD} = 5 V ±5%			
V_{OLT}	TX0 and TX1 output voltage LOW	I_O = 1.2 mA; note 3	–	0.1	V
		I_O = 10 mA	–	1.0	V
V_{OHT}	TX0 and TX1 output voltage HIGH	I_O = 1.2 mA; note 3	V_{DD} – 0.1	–	V
		I_O = 10 mA	V_{DD} – 1.0	–	V

Notes

1. (AD0 – AD7) = ALE = $\overline{RD}$ = $\overline{WR}$ = $\overline{CS}$ = V_{DD}; MODE = V_{SS}; RX0 = 2.7 V; RX1 = 2.3 V; XTAL1 = 0.5V/V_{DD} – 0.5 V; all outputs unloaded.
2. (AD0 – AD7) = ALE = $\overline{RD}$ = $\overline{WR}$ = $\overline{INT}$ = $\overline{RST}$ = $\overline{CS}$ = MODE = RX0 = V_{DD}; RX1 = XTAL1 = V_{SS}; all outputs unloaded.
3. Not tested during production.

12.0 AC CHARACTERISTICS

V_{DD} = 5 V ±10%; V_{SS} = 0 V; C_L = 50 pF (output pins); T_{amb} = −40 °C to + 125 °C; unless otherwise specified (note 1)

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
f_{CLK}	Oscillator frequency		3	16	MHz
t_{SU1}	Address set-up to ALE/AS LOW		10	-	ns
t_{HD1}	Address hold time		22	-	ns
t_{PW1}	ALE/AS pulse width		60	-	ns
t_{VD1}	$\overline{RD}$ LOW to valid data output	Intel mode	-	148	ns
t_{VD2}	E HIGH to valid data output	Motorola mode	-	148	ns
t_{DF1}	Data float after $\overline{RD}$ HIGH	Intel mode	10	55	ns
t_{DF2}	Data float after E LOW	Motorola mode	10	55	ns
t_{SU2}	Input data set-up to $\overline{WR}$ HIGH	Intel mode	30	-	ns
t_{HD2}	Input data hold after $\overline{WR}$ HIGH	Intel mode	13	-	ns
t_{HH1}	WR HIGH to next ALE/AS HIGH		23	-	ns
t_{SU3}	Input data set-up to E LOW	Motorola mode	30	-	ns
t_{HD3}	Input data hold after E LOW	Motorola mode	25	-	ns
t_{LL1}	ALE LOW to $\overline{WR}$ LOW	Intel mode	10	-	ns
t_{LL2}	ALE LOW to $\overline{RD}$ LOW	Intel mode	10	-	ns
t_{LH1}	AS LOW to E HIGH	Motorola mode	10	-	ns
t_{SU4}	Set-up time of RD/$\overline{WR}$ to E HIGH	Motorola mode	20	-	ns
t_{PW2}	$\overline{WR}$ pulse width	Intel mode	170	-	ns
t_{PW3}	$\overline{RD}$ pulse width	Intel mode	170	-	ns
t_{PW4}	E pulse width	Motorola mode	170	-	ns
t_{LL3}	$\overline{CS}$ LOW to $\overline{WR}$ LOW	Intel mode	0	-	ns
t_{LL4}	$\overline{CS}$ LOW to $\overline{RD}$ LOW	Intel mode	0	-	ns
t_{LH2}	$\overline{CS}$ LOW to E HIGH	Motorola mode	0	-	ns
Input comparator/output driver					
t_{sd}	Sum of the input and output delays	V_{DD} = 5 V ±5% V_{DIF} = ±42 mV 1.4 V < V_I < V_{DD} - 1.4 mV	-	62	ns

Note to the AC characteristics

1. AC characteristics are not tested.

12.1 AC timing diagrams

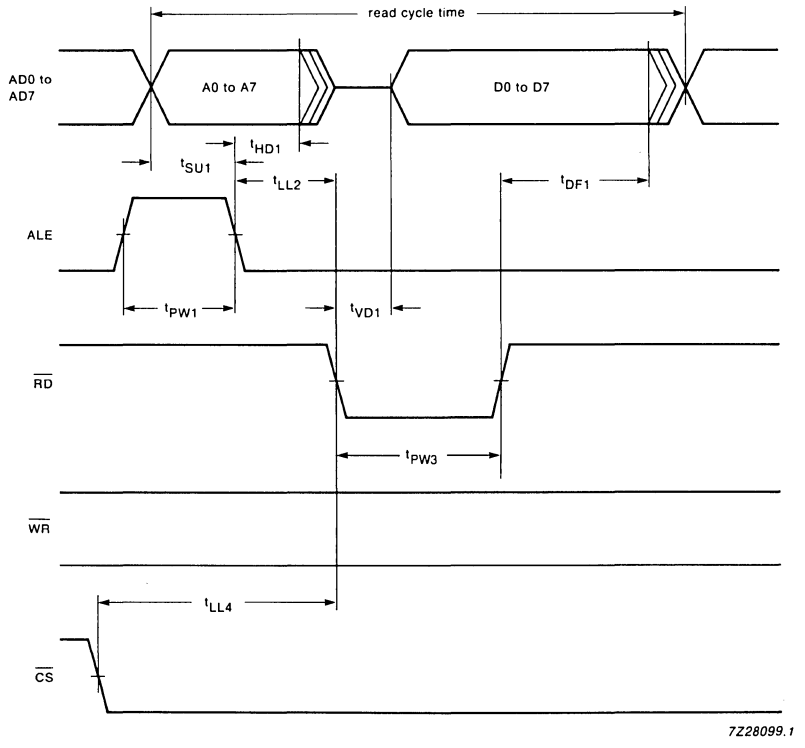


Fig.12.1 Read cycle timing diagram; Intel mode.

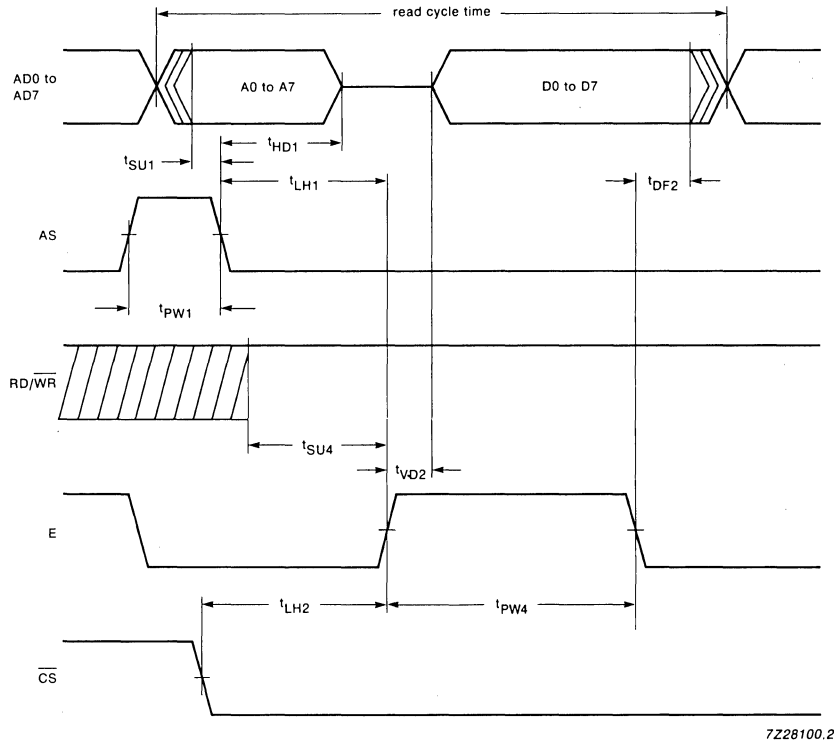


Fig.12.2 Read cycle timing diagram; Motorola mode.

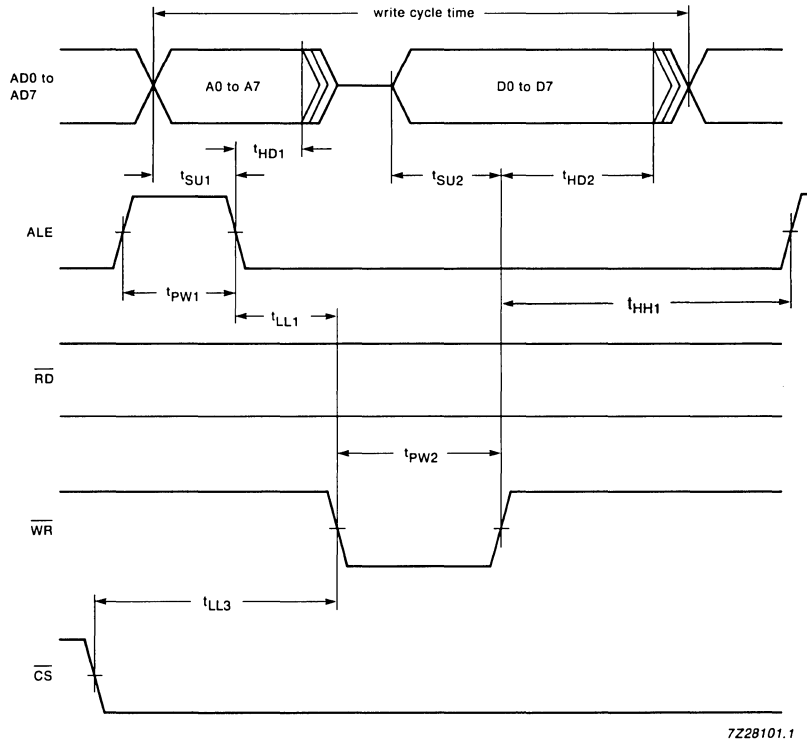


Fig.12.3 Write cycle timing diagram; Intel mode.

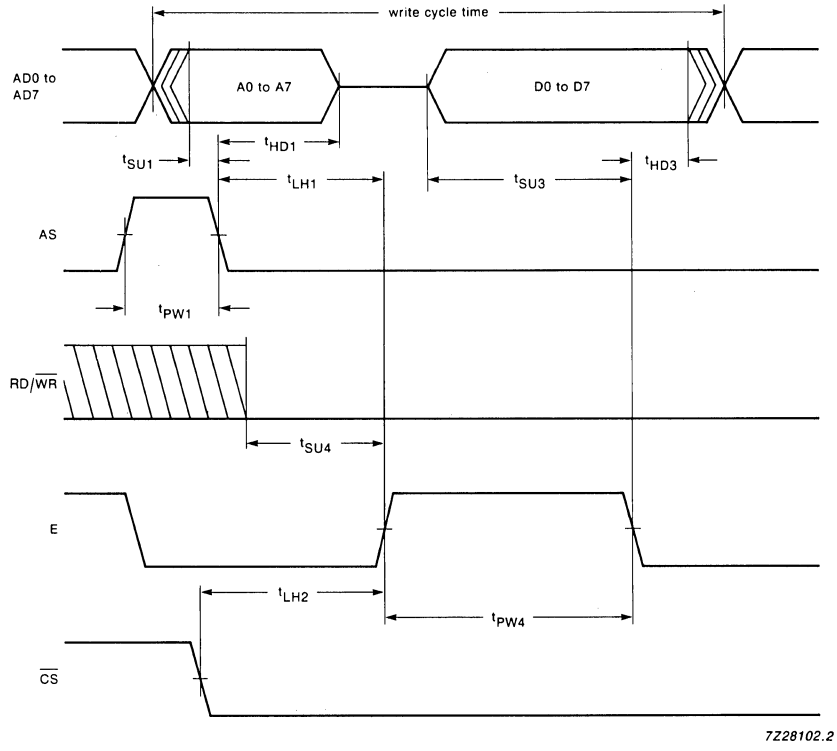


Fig.12.4 Write cycle timing diagram; Motorola mode.

12.2 Additional AC information

To provide optimum noise immunity under worse case conditions, the chip is powered by three separate pins and grounded by three separate pins, see Fig.12.5.

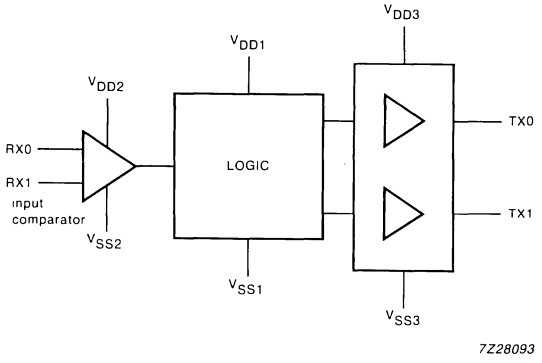


Fig.12.5 Optimized noise immunity block diagram.

Voltage regulator with watchdog for microprocessor/controller systems

UAA1300

FEATURES

- Driving circuit for external PNP power transistor with adjustable output voltage via sense path, short-circuit protected
- Additional 5 V/50 mA output for RAM buffering, short-circuit protected
- Operating voltage range: 5.7 V to 24 V
- Reset with adjustable trigger pulse length activated by output voltage < 4.6 V
- Watchdog with adjustable input trigger window
- Dual polarity reset and watchdog output pulse
- Low line detection adjustable by external resistor ratio
- Enable input to activate watchdog and driving circuit
- Low quiescent current typical: 380 µA
- Thermal protection

GENERAL DESCRIPTION

The UAA1300 is a bipolar IC voltage regulator especially designed for use within an automotive environment and also suitable to provide enhanced facilities within many microcontroller applications. The UAA1300 provides two stabilized low-drop outputs and offers special control functions to increase system protection and reliability.

QUICK REFERENCE DATA

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
V_{DD}	supply voltage		+5.7	-	+24	V
I_Q	quiescent current		-	380	-	µA
I_{sb}	standby output current		50	-	-	mA
V_{sb}	standby output voltage		+4.9	-	+5.1	V
L	load regulation		-	1	-	mV/mA
$I_{O(p)}$	driving current external PNP		10	-	-	mA
G	transconductance (sense input)		-	+1.5	-	mA/mV
T_{amb}	operating ambient temperature range SOT27 SOT163A	V_{DD} = 13 V I_{sb} = 50 mA	 −40 −40	 - -	 +130 +85	 °C °C
P_{tot}	total power dissipation				see Fig.7	

ORDERING INFORMATION

EXTENDED TYPE NUMBER	PACKAGE			
	PINS	PIN POSITION	MATERIAL	CODE
UAA1300	14	DIL	plastic	SOT27
UAA1300T	20	mini-pack	plastic	SOT163A

Voltage regulator with watchdog for microprocessor/controller systems

UAA1300

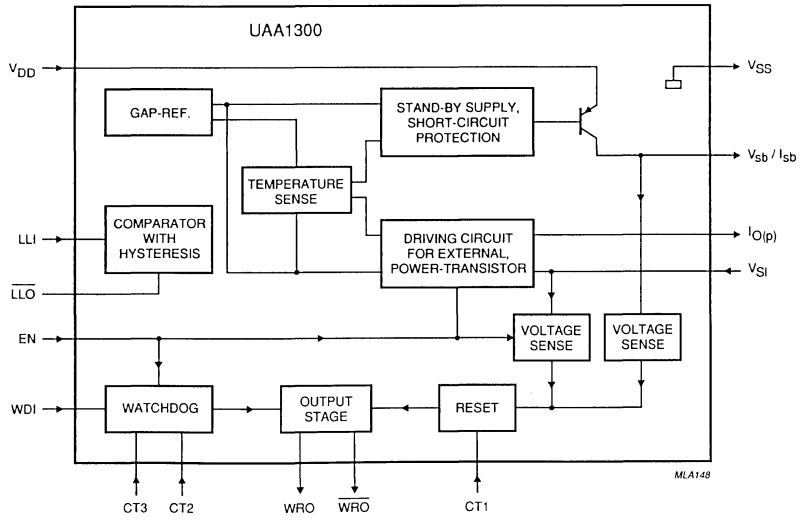


Fig.1 Block diagram.

Voltage regulator with watchdog for microprocessor/controller systems

UAA1300

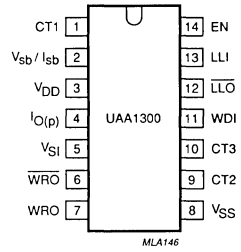


Fig.2(a) Pinning diagram; SOT27.

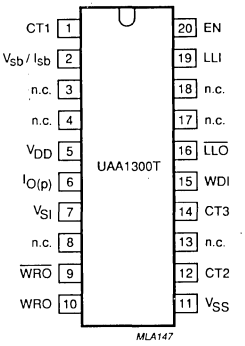


Fig.2(b) Pinning diagram; SOT163A.

PINNING

SYMBOL	PIN		DESCRIPTION
	SOT27	SOT163A	
CT1	1	1	reset-length
V_{sb}/I_{sb}	2	2	standby output: 5 V/50 mA
V_{DD}	3	5	supply voltage (line)
$I_{O(p)}$	4	6	driving current external pnp
V_{SI}	5	7	sense input external pnp
$\overline{WRO}$	6	9	watchdog reset output active LOW
WRO	7	10	watchdog reset output active HIGH
V_{SS}	8	11	ground
CT2	9	12	maximum watchdog period
CT3	10	14	minimum watchdog period
WDI	11	15	watchdog input
$\overline{LLO}$	12	16	low line output
LLI	13	19	low line input
EN	14	20	enable input

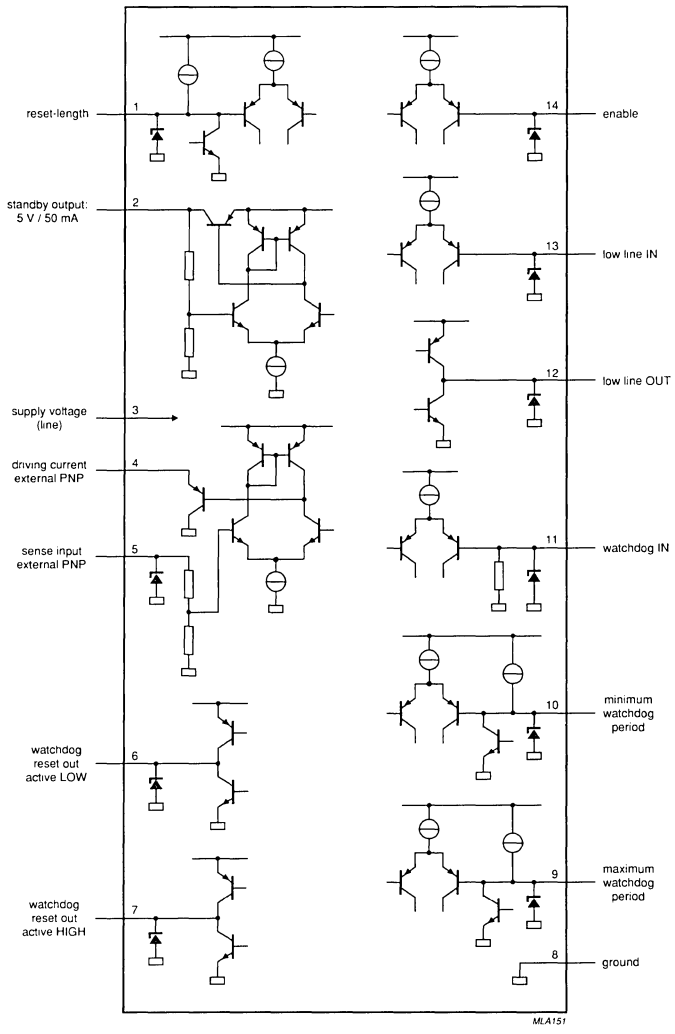


Fig.3 Internal interfaces.

Voltage regulator with watchdog for microprocessor/controller systems

UAA1300

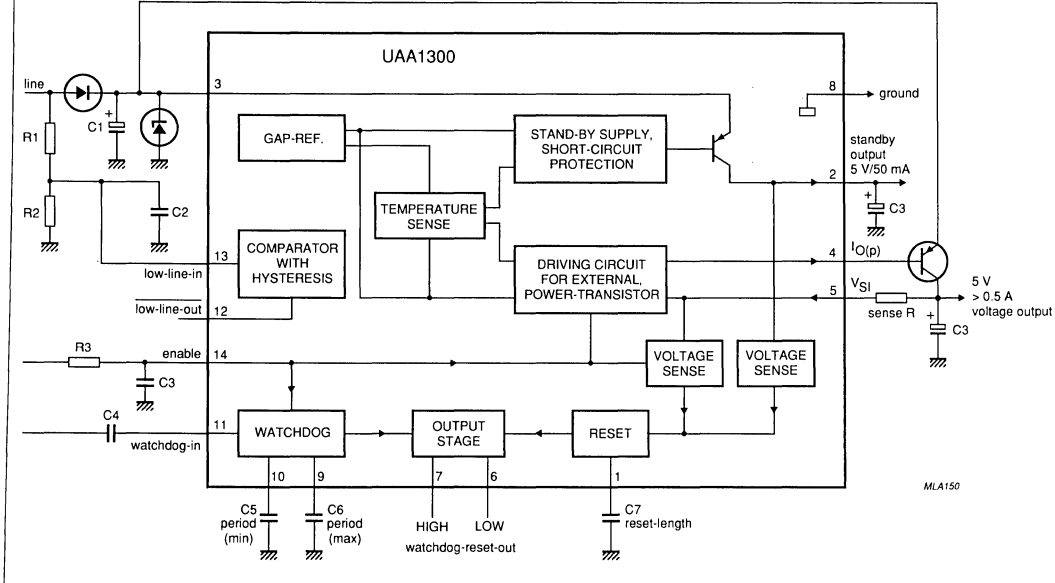


Fig.4 Functional diagram.

Voltage regulator with watchdog for microprocessor/controller systems

UAA1300

FUNCTIONAL DESCRIPTION

The following features of the UAA1300 are described below:

- operating voltage range
- outputs
- $\overline{WRO}$/WRO activated by low output voltage
- $\overline{WRO}$/WRO activated by WDI pulse timing
- LLI detection
- enable input
- thermal protection

Operating voltage range

The UAA1300 operates within a supply voltage range of +5.7 V to +24 V and provides reverse polarity protection and low line voltage detection.

Outputs

Two stabilized short-circuit protected; low drop outputs are provided:

Output current $I_{O(p)}$ drives an external PNP power transistor when switched on by the enable input (EN). Voltage sense input (V_{SI}) regulates the current at the external power transistor. With R = 0 Ω the voltage is 5 V ± 100 mV. Using a resistor > 0 Ω the output will be regulated to a voltage > 5 V (see Figs 4, 16, 17).

A fixed 5 V/50 mA output V_{sb}/I_{sb} supplies for example, storage circuits in microprocessor/controller systems. Not switched by enable input (EN).

$\overline{WRO}$/WRO activated by low output voltage

When either power or standby output voltage (pinning references $I_{O(p)}$, V_{sb}/I_{sb}) < 4.6 V, the watchdog output stage is triggered by a signal from the voltage sense stages and forces a static watchdog reset output pulse ($\overline{WRO}$/WRO) when both the power and standby output voltage rise above 4.7 V. The duration of the pulse width is determined by the value of external capacitor C7 (see Figs 5 and 18). To operate with external series control transistor, the watchdog function must first be switched on by the enable input (EN).

$\overline{WRO}$/WRO activated by watchdog input pulse (WDI) timing

The watchdog input pulse (WDI) from the µC/µP must be received within a time window determined by external capacitors C5 and C6. When this is not achieved the watchdog output stage generates a $\overline{WRO}$/WRO pulse with fixed width of 40 µs. Each rising edge at WDI restarts the timing cycle (see Figs 6 and 19). To operate, this watchdog function must first be switched on by the enable input (EN).

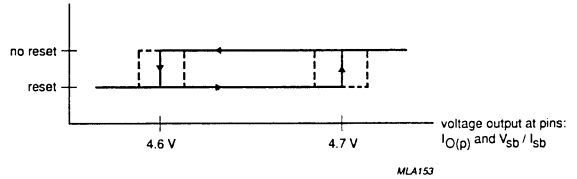


Fig.5 Typical values; threshold voltage (falling) reset/no reset timing parameters.

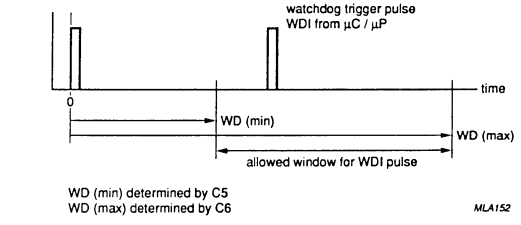


Fig.6 Watchdog timing window.

LLI detection

The UAA1300 provides detection for low line input (LLI).
This is achieved by use of a comparator which generates a low line output pulse ($\overline{LLO}$) on detection of a low line input voltage (LLI) < 1.24 V. When the LLI < 1.24 V it is considered LOW (V). When the LLI > 1.34 V it is considered HIGH (V). The threshold voltage is adjusted by the external resistor ratio of R1/R2.

An additional external capacitor C2 prevents an initiation of an $\overline{LLO}$ pulse which might otherwise be caused by short duration (transient) LLI voltage drops (see Fig.4).

Enable input

When the Enable input (EN) is > 2 V the driving circuit for the external power transistor is switched on and the watchdog functions are activated.

Thermal protection

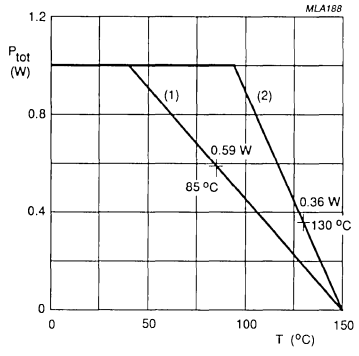
Outputs (pinning references $I_{O(p)}$ and V_{sb}/I_{sb}) are controlled by an internal temperature sense switch module which operates at a crystal temperature > 150 °C switching off these outputs. The temperature sense switch resets at a crystal temperature ≤ +130 °C and switches on both outputs.

LIMITING VALUES

In accordance with the Absolute Maximum System (IEC 134).

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
T_{amb}	operating ambient temperature range SOT27 SOT163A	V_{DD} = 13 V I_{sb} = 50 mA	 −40 −40	 - -	 +130 +85	 °C °C
P_{tot}	total power dissipation				see Fig.7	
T_{stg}	storage temperature range		−55	-	+150	°C
V_{DD}	supply voltage (reverse polarity protected)		−25	-	+25	V
V_{DD}	supply voltage	t = 2 ms	-	-	+30	V
V_I	input voltages (all inputs)		−0.5	-	+7	V
I_I	input current (all inputs)		−5	-	5	mA
V_O	output voltages (LLO, WRO)		-	-	+7	V
I_O	output current (LLO, WRO)		-	-	10	mA
$V_{O(p)}$	output voltage (power)		−0.5	-	V_{DD}	V
$I_{O(p)}$	output current (power)		-	-	25	mA
V_{sb}	output voltage (standby)		-	-	V_{DD}	V
I_{sb}	output current (standby)		-	-	100	mA
Dynamic limits; transient voltage:						
V_{ES}	electrostatic handling *	V_{ES} for all pins	−2000	-	+2000	V

* Electrostatic handling is equivalent to discharging a 100 pF capacitor through a 1.5 kΩ series resistor with a 15 ns rise time.



(1) = SO20; SOT163A

(2) = SOT27 (IDF14)

$P_{tot} \approx (V_{DD} - 5\ V) \times I_{sb} + V_{DD} \times I_D$

Fig.7 Power derating diagram.

CHARACTERISTICS

V_{DD} = 13 V, T_{amb} = 25 °C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
V_{DD}	supply voltage range for logic functions		+4.2	-	+24	V
I_{DD}	supply current without load	input EN = LOW	-	380	-	µA
T_{amb}	operating ambient temperature range		−40	-	+130	°C
Standby output						
V_{sb}	output voltage	V_{DD} min = +5.7 V	+4.9	+5	+5.1	V
I_{sb}	current capability		−50	-	-	mA
L	load regulation		-	-	1	mV/mA
TC_{sb}	temperature coefficient of output voltage		-	±400	-	µV/°C
RR	ripple rejection	f = 100 Hz	-	50	-	dB
Reference for external pnp-transistor						
I_O	current capability		10	-	-	mA
V_O	limits for external voltage		+5	-	V_{DD}	V
G	transconductance (sense input)		0.4	1.5	-	mA/mV
I_S	sense current	V_{SI} = +5V	37.5	50	62.5	µA
R_S	input resistance		75	100	125	kΩ
Temperature sense						
δS	threshold for rising temperature		-	+150	-	°C
δΔ	hysteresis of temperature sense		-	+30	-	°C

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
Low-line detection						
$I_{LLI/LOW}$	input current LOW		-	-	−0.2	µA
V_{THL}	threshold voltage for falling voltage	temperature range −40 °C to +130 °C	+1.22	+1.24	+1.26	V
V_{THU}	hysteresis		+90	+100	+110	mV
T_C	temperature coefficient for hysteresis		-	+300	-	µV/°C
Low-line output						
V_{OL}	output voltage LOW	I_{OL} = 2 mA	-	-	+0.4	V
V_{OH}	output voltage HIGH	I_{OH} = 50 µA	+3.4	-	-	V
Enable input						
V_{IL}	enable input voltage LOW		-	-	+0.8	V
I_{IL}	enable input current LOW		-	-	−0.1	µA
V_{IH}	enable input voltage HIGH		+2	-	-	V
Watchdog input						
V_{LW}	watchdog input voltage LOW		-	-	+0.8	V
I_{LW}	watchdog input current LOW		-	-	−0.2	µA
V_{HW}	watchdog input voltage HIGH		+2	-	-	V
Z_I	input impedance		28	50	72	kΩ
t_{WDI}	minimum input pulse length		40	-	-	µs
Minimum and maximum Watchdog Period						
I_{WD}	output current (pin shorted to ground)		−1	−1.25	−1.50	µA
t_{CIWD}	temperature coefficient of output current		-	−0.6	-	‰/°C
V_{TH}	threshold for internal comparator		-	+1.24	-	V
Timing of watchdog function Range of useful external film-capacitors: 100 pF to 1 µF (C5/C6)						
t_{WP}	watchdog minimum period	C5 = 100 pF	70	100	130	µs
t_{WP}	watchdog maximum period	C6 = 1 µF	700	1000	1300	ms
t_{WDO}	duration of the watchdog output pulse		25	40	55	µs
Reset function: timing						
t_{RO}	duration of the reset pulse	C7 = 100 pF	-	100	-	µs
t_{RO}	duration of the reset pulse	C7 = 100 nF	-	100	-	ms
Reset function: DC conditions						
V_{th}	threshold voltage (falling) for reset on		4.5	4.6	4.7	V
δ	hysteresis		-	100	-	mV
T_C	temperature coefficient of hysteresis		-	100	-	µV/°C

TEST VALUES, ADJUSTMENTS AND APPLICATION EXAMPLE

The following figures (Figs 8 to 19) provide a test circuit, typical test values and voltage and timing adjustments. Figure 20 provides an application example for automotive environment.

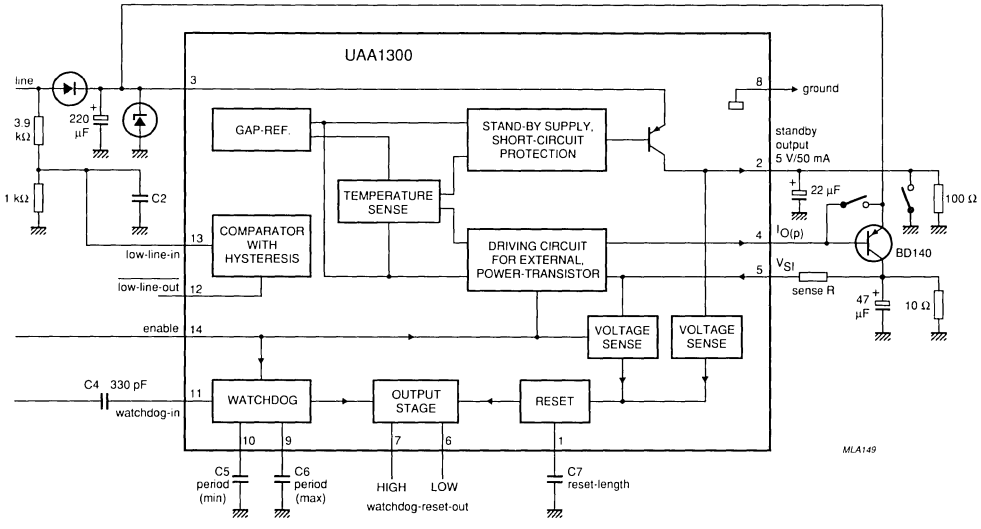


Fig.8 Test circuit.

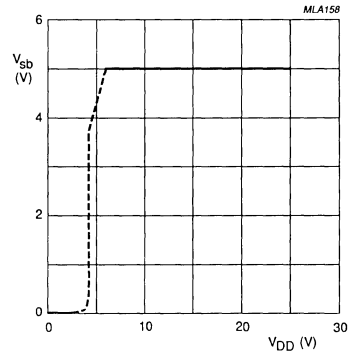


Fig.9 Standby output voltage (V_{sb}) as a function of line voltage (V_{DD}); typical values; measured in Fig.8.

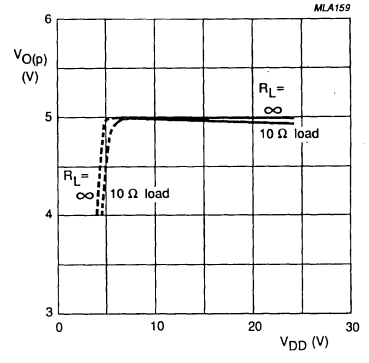


Fig.10 Power output voltage ($V_{O(p)}$) as a function of line voltage (V_{DD}); typical values; measured in Fig.8.

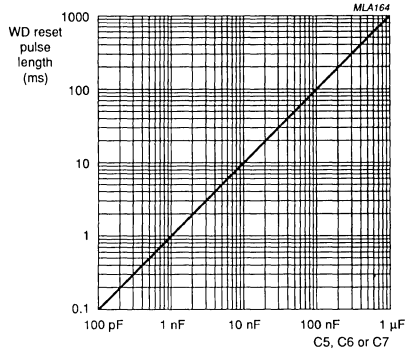


Fig.11 Watchdog reset pulse length as a function of external capacitors C5 or C6 or C7.

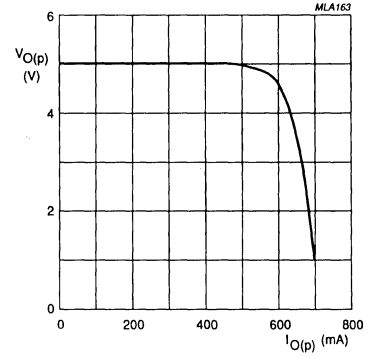


Fig.12 Typical power output voltage ($V_{O(p)}$) as a function of power output current ($I_{O(p)}$); V_{DD} = 7 V; behaviour is influenced by externally chosen transistor type.

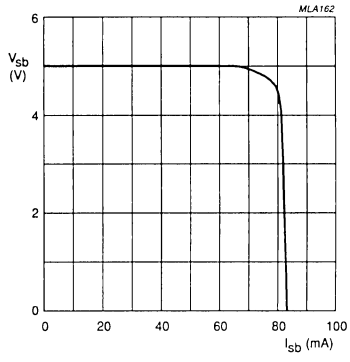


Fig.13 Typical standby output voltage (V_{sb}) as a function of standby output current (I_{sb}); V_{DD} = 7 V.

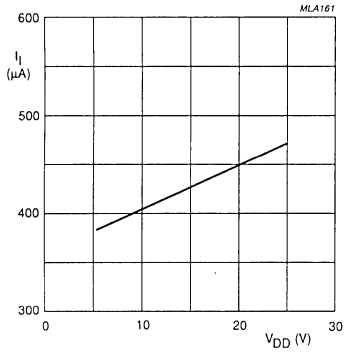


Fig.14 Quiescent current (I_I) as a function of line voltage (V_{DD}); no load at standby output voltage (V_{sb}); enable (EN) not active.

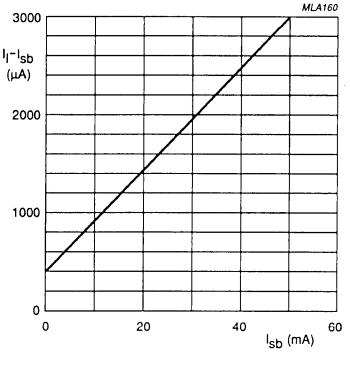


Fig.15 Quiescent current (I_I) minus standby output current (I_{sb}) as a function of standby output current (I_{sb}); V_{DD} = 13 V.

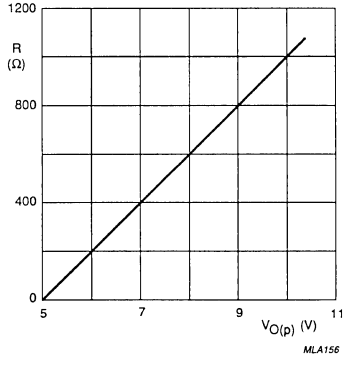


Fig.16 Adjustment of power output voltage ($V_{O(p)}$); typical values of external adjustment resistors: circuit with high accuracy.

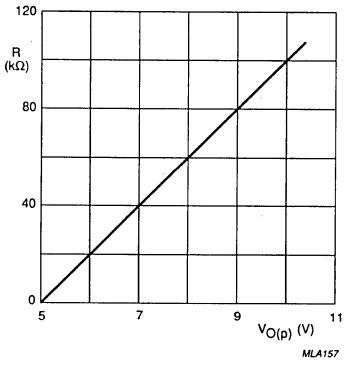


Fig.17 Adjustment of power output voltage ($V_{O(p)}$); typical values of external adjustment resistors: circuit with less accuracy but reduced current consumption of the application.

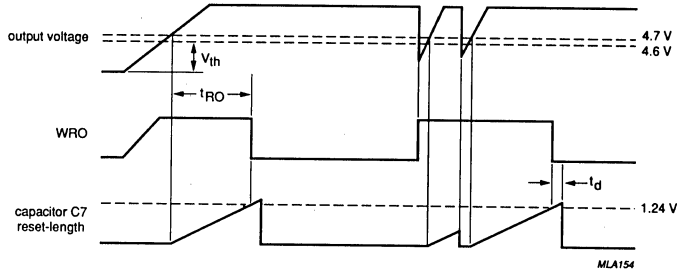
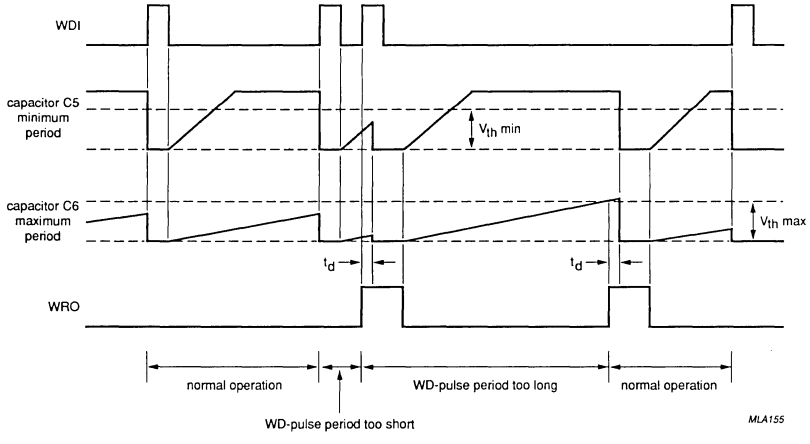


Fig.18 Reset behaviour: watchdog reset output (WRO) timing as a function of output voltage (power $V_{O(p)}$ or standby V_{sb}) with external capacitor C7 as a parameter.



V_{th}min = threshold voltage for WD(min)

V_{th}max = threshold voltage for WD(max)

t_d = internal delay

normal operation: V_{th}min must already be achieved

V_{th}max must not be achieved

if these conditions are not present a WDI pulse will force a $\overline{WRO}$/WRO pulse at the output stage

Fig.19 Reset behaviour: watchdog reset output (WRO) timing as a function of watchdog input (WDI) with external capacitors C5 and C6 as parameters.

APPLICATION INFORMATION

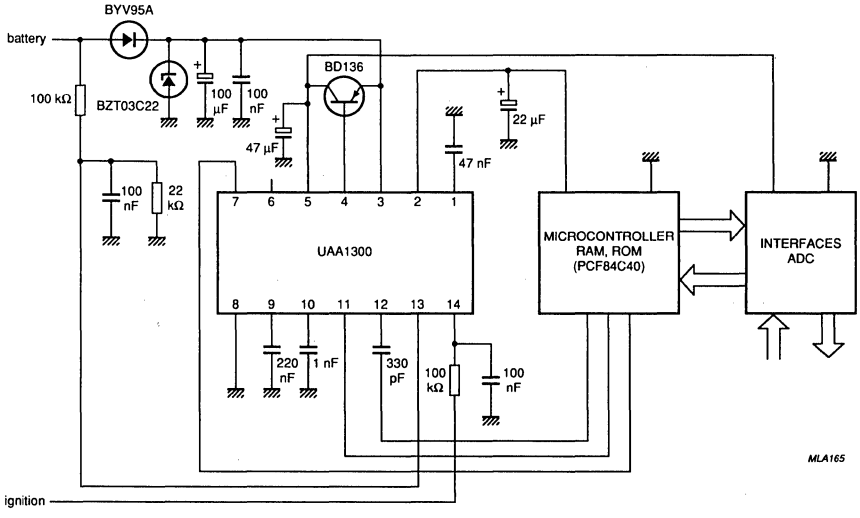


Fig.20 Application example for automotive environment.

Power failure detector and reset generator — PCF1252-X FAMILY

GENERAL DESCRIPTION

The PCF1252-X family are CMOS voltage detectors designed especially for power-ON/OFF detection in microcontroller/microprocessor systems (for initialization and data storage purposes). The output $\overline{POWF}$ is activated at a precise, temperature stable, trip-point. The RESET output has a built-in delay with duration determined by an external capacitor (C_{CT}). A second comparator (comparator 2) has been included to allow for the possibility of a second monitoring point in the system.

Features

- Low current consumption, typically 6 μA
- 10 versions available, trip-points vary from 2.55 V to 4.75 V
- Temperature stable trip-point
- Variable RESET delay
- Reset polarity selection
- Comparator for second level detection (e.g. overvoltage detection)
- Advance warning of power failure

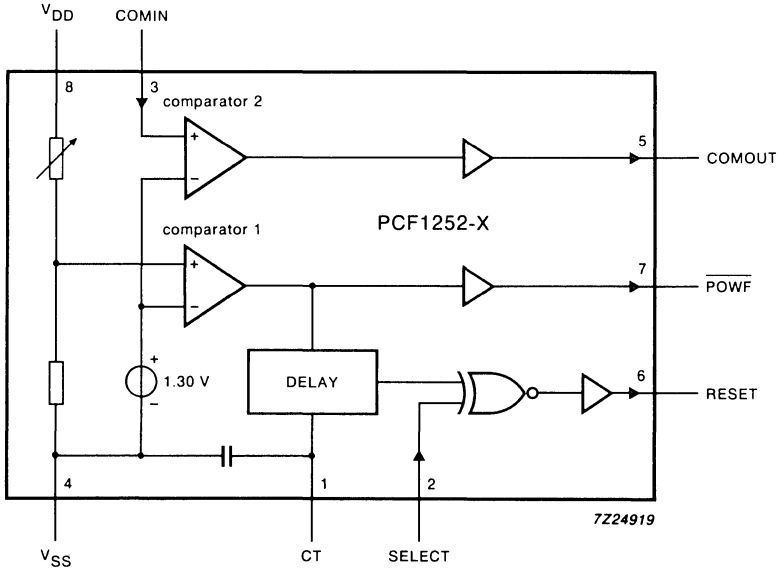


Fig.1 Block diagram.

PACKAGE OUTLINES

PCF1252-XP: 8-lead DIL; plastic (SOT97).
PCF1252-XT: 8-lead mini-pack; plastic (SO8; SOT96A).

PINNING

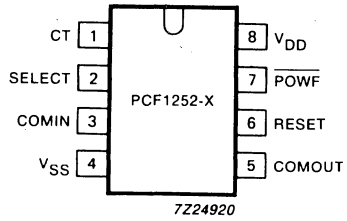


Fig.2 Pinning diagram.

pin no.	mnemonic	description
1	CT	connection for the external capacitor
2	SELECT	select polarity or external reset input
3	COMIN	comparator input
4	V_{SS}	ground (0 V)
5	COMOUT	comparator output
6	RESET	reset output
7	$\overline{POWF}$	power failure signal output
8	V_{DD}	positive supply voltage

FUNCTIONAL DESCRIPTION (see Fig.1)

The PCF1252-X contains a precise factory-programmed voltage reference, two comparators and a delay circuit. The PCF1252-X family is comprised of 10 versions with varying voltage trip-points (V_{TRIP}), see section "Characteristics".

Supply

The supply voltage (V_{DD}) is internally divided before being compared, via comparator 1, with the internal reference voltage.

$\overline{POWF}$ (see Fig.3)

The $\overline{POWF}$ output is:

- LOW, if V_{DD} is below V_{TRIP}.
- HIGH, if V_{DD} is above V_{TRIP}.

Power-ON reset (SELECT = LOW)

As V_{DD} rises past V_{TRIP}, a positive reset pulse is generated at RESET. The duration of the reset pulse (t_R) is determined by the value of the external capacitor (C_{CT}; maximum 1 μF, see Fig.8) connected to CT. With no external capacitor connected, C_{CT} assumes a minimum value of 100 pF. If SELECT is HIGH, the reset pulse is inverted.

Power failure

During a power-OFF condition ($V_{DD} < V_{TRIP}$), $\overline{POWF}$ goes LOW. After a time delay (t_S), also determined by C_{CT}, RESET goes HIGH. Any $\overline{POWF}$ output ($V_{DD} < V_{TRIP}$) will result in a subsequent RESET pulse.

Voltage trip-point

By selecting the voltage trip-point slightly higher than the minimum operating voltage of the microcontroller/microprocessor, there is sufficient time for data storage before the power actually fails.

In order to prevent oscillations around the voltage trip-point, a small hysteresis has been included, resulting in a power-ON switching point that is higher than the voltage trip-point (minimum of 15 mV). The voltage trip-point refers to the value at which power-OFF is signalled.

COMIN

Input to the second comparator (comparator 2). When used in conjunction with an external voltage divider, this allows a second point in the system to be monitored. This input has no built-in hysteresis. When not in use connect to V_{DD}. COMOUT will be LOW or HIGH depending on the voltage at COMIN:

- COMOUT = HIGH, if voltage at COMIN is above the switch point V_{SP} (typically 1.30 V).
- COMOUT = LOW, if voltage at COMIN is below the switch point V_{SP} (typically 1.30 V).

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

parameter	conditions	symbol	min.	max.	unit
Supply voltage range		V_{DD}	−0.5	+ 7	V
Input voltage range		V_I	−0.5	V_{DD} + 0.5	V
DC clamp-diode current	all pins; $V_I < -0.5$ V or $> V_{DD}$ + + 0.5 V	I_I	–	20	mA
Output current		I_O	–	20	mA
Total power dissipation		P_{tot}	–	150	mW
Storage temperature range		T_{stg}	−65	+ 100	°C
Operating ambient tempetature range		T_{amb}	−40	+ 85	°C

HANDLING

Inputs and outputs are protected against electrostatic discharge in normal handling. However, to be totally safe, it is desirable to take normal handling precautions appropriate to handling MOS devices (see 'Handling MOS Devices').

CHARACTERISTICS (see Fig.3)
V_{DD} = 2.4 V to 6.0 V; V_{SS} = 0 V; T_{amb} = −40 °C to + 85 °C; unless otherwise specified

parameter	conditions	symbol	min.	typ.	max.	unit
Supply voltage range		V_{DD}	2.4	–	6.0	V
Voltage trip-point:	T_{amb} = + 25 °C					
PCF1252-0		V_{TRIP}	4.70	4.75	4.80	V
PCF1252-1		V_{TRIP}	4.50	4.55	4.60	V
PCF1252-2		V_{TRIP}	4.20	4.25	4.30	V
PCF1252-3		V_{TRIP}	4.00	4.05	4.10	V
PCF1252-4		V_{TRIP}	3.70	3.75	3.80	V
PCF1252-5		V_{TRIP}	3.50	3.55	3.60	V
PCF1252-6		V_{TRIP}	3.20	3.25	3.30	V
PCF1252-7		V_{TRIP}	3.00	3.05	3.10	V
PCF1252-8		V_{TRIP}	2.70	2.75	2.80	V
PCF1252-9		V_{TRIP}	2.50	2.55	2.60	V
Supply current	T_{amb} = + 25 °C; see Figs 4 and 5 V_{DD} = V_{TRIP} + 0.5 V; COMIN = V_{DD}	I_{DD}	–	6	10	μA
Voltage trip-point temperature coefficient	note 1	ΔV_{TRIP}	–	±100 x 10^{-6}	±400 x 10^{-6}	/K
Voltage trip-point hysteresis		V_H	15	30	50	mV
COMIN switch point	T_{amb} = + 25 °C	V_{SP}	1.28	1.30	1.32	V
COMIN switch point temperature coefficient	note 1	ΔV_{SP}	–	±0.1	±0.5	mV/K
SELECT input voltage:						
LOW		V_{IL}	–	–	$0.3V_{DD}$	V
HIGH		V_{IH}	$0.7V_{DD}$	–	–	V
SELECT and COMIN leakage current:						
LOW		$-I_{IL}$	–	–	1.0	μA
HIGH		I_{IL}	–	–	1.0	μA
$\overline{\text{POWF}}$, RESET and COMOUT						
Output sink current	see Fig.6; V_O = 0.4 V; V_{DD} = 2.4 V	I_O	1	3	–	mA
Output source current	see Fig.7; V_O = 2.0 V; V_{DD} = 2.4 V	$-I_O$	0.75	2	–	mA
Reset time	note 2; C_{CT} = 1 nF	t_R	400	1000	2000	μs
Save time	note 2; C_{CT} = 1 nF	t_S	40	100	200	μs
Reset to save time ratio		t_R/t_S	–	10	–	

CHARACTERISTICS (continued)

parameter	conditions	symbol	min.	typ.	max.	unit
CT internal capacitance		C_{INT}	–	100	–	pF

Notes to the characteristics

1. Value given per degree Kelvin. Tested on a sample basis.
2. Conformance to these specifications is only guaranteed when the slew rate of V_{DD} is less than 25 V/ms.

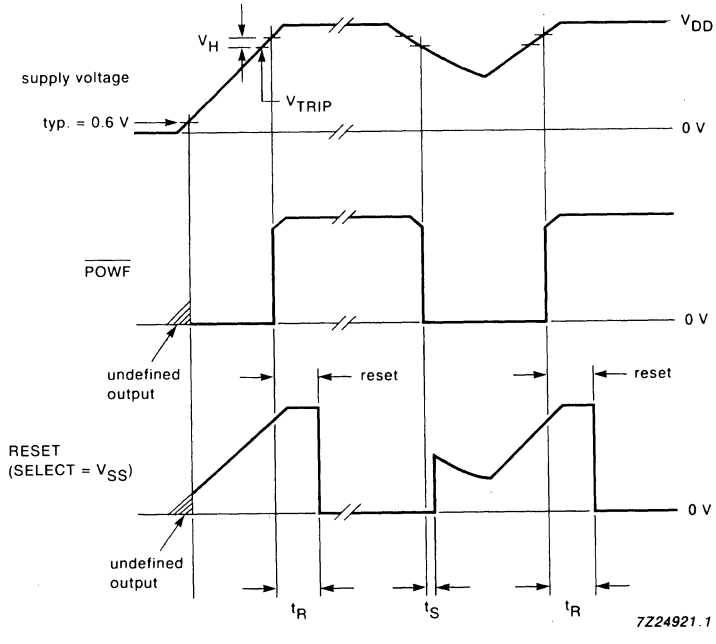


Fig.3 Timing diagram.

Typical performance characteristics

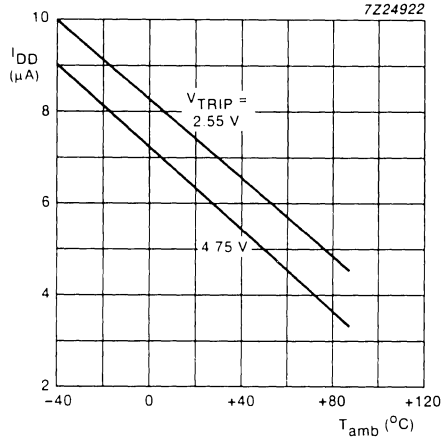


Fig.4 Supply current as a function of temperature; V_{DD} = 5 V; COMIN = V_{DD}.

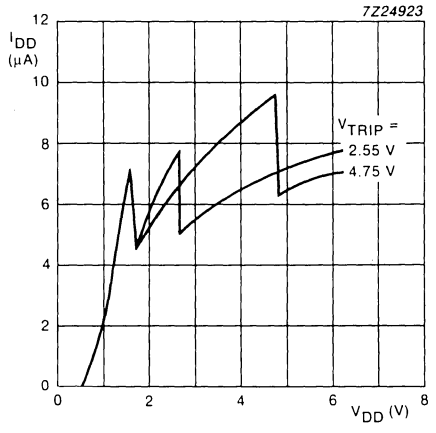


Fig.5 Supply current as a function of the supply voltage; T_{amb} = + 25 °C.

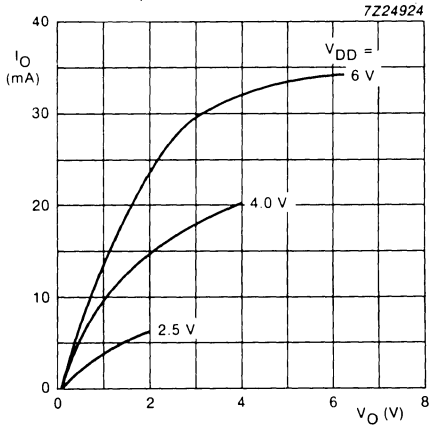


Fig.6 Output sink current as a function of the output voltage.

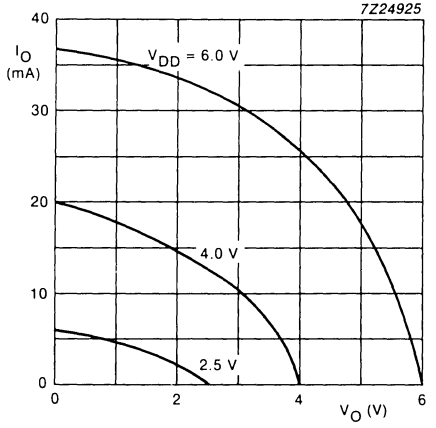


Fig.7 Output source current as a function of the output voltage.

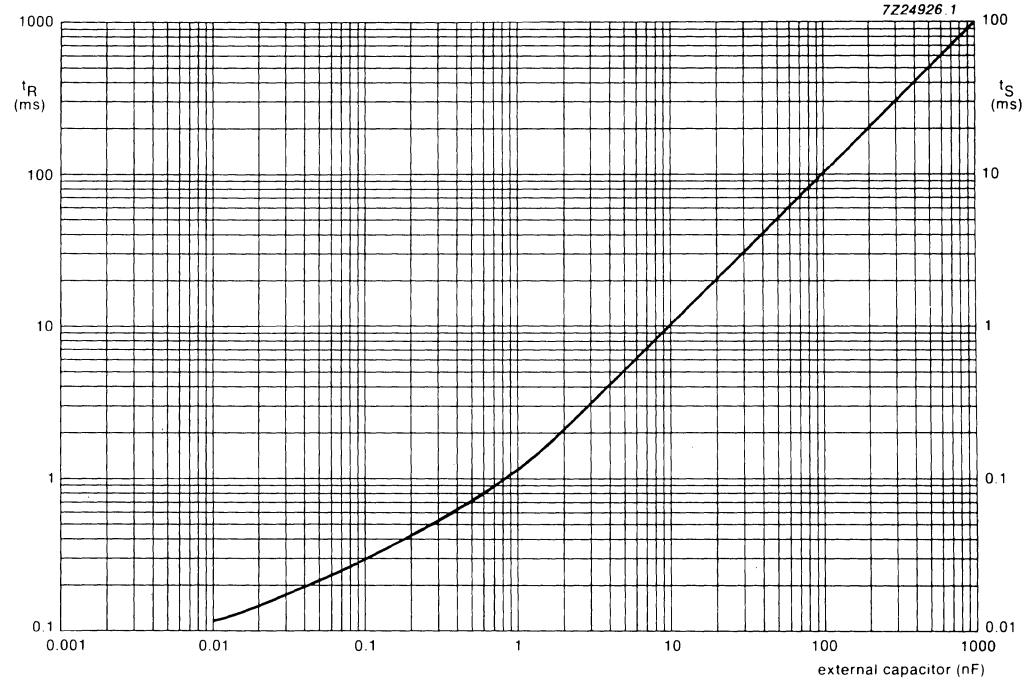


Fig.8 Reset and save times as a function of the external capacitor (C_{CT}).

Notes to Fig.8

1. t_R (typ.) = (0.1 + C_{CT}) ms.
2. t_S (typ.) = (0.01 + 0.1C_{CT}) ms.

APPLICATION INFORMATION

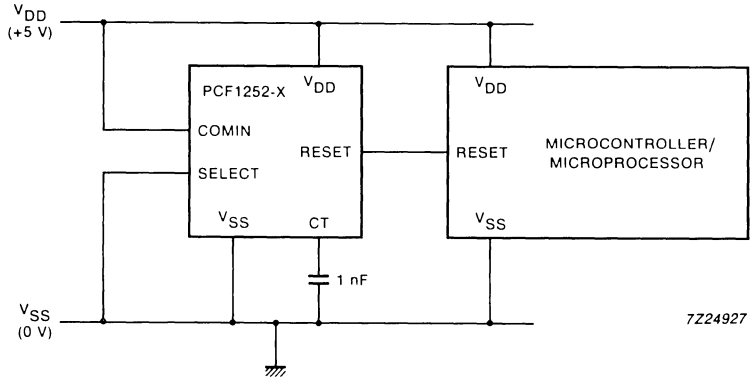


Fig.9 Typical power-ON reset circuit for a microcontroller/microprocessor system; (when not used, COMIN must be connected to V_{DD}).

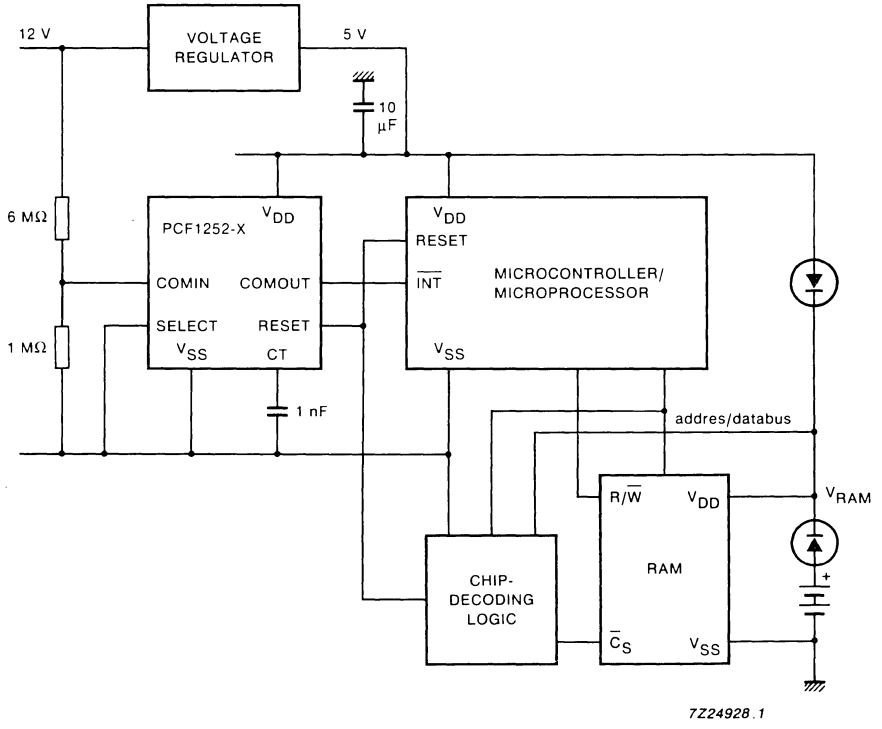


Fig.10 Data retention circuit for memory back-up systems.

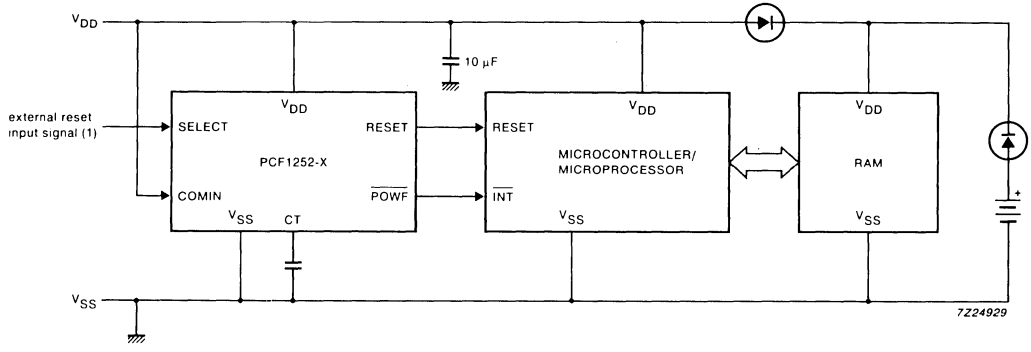


Fig.11 Data retention circuit with external switchable reset for systems with a single voltage supply.

Note to Fig.11

1. For external reset application, the SELECT input must be debounced.

Section 8
Package Outlines

80C51-Based
8-Bit Microcontrollers

CONTENTS

24-Pin (300 mils wide) Ceramic Dual In-Line with Quartz Window (F) Package 981

24-Pin (300 mils wide) Plastic Dual In-Line (N) Package 982

28-Pin (600 mils wide) Ceramic Dual In-Line with Quartz Window (F) Package 983

28-pin (600 mils wide) Plastic Dual In-Line (N) Package 984

SOT117 28-Pin Plastic Dual In-Line (N/P) Package 985

28-Pin Plastic Leaded Chip Carrier (A) Package 986

SOT136A 28-Pin Plastic SO (Small Outline) Dual In-Line (D/T) 987

40-Pin (600 mils wide) Ceramic Dual In-Line with Quartz Window (F/FA) Package 988

40-Pin (600 mils wide) Plastic Dual In-Line (N) Package 989

SOT129 40-Pin Plastic Dual In-Line (P/N) Package 990

SOT158A 40-Pin Plastic VSO (Very Small Outline) Dual In-Line (D/T) Package 991

44-Pin Plastic Leaded Chip Carrier (A) Package 992

44-Pin Square Ceramic Leaded Chip Carrier with Quartz Window (L) Package 993

44-Pin Cerquad J-Bend with Quartz Window (K/KA) Package 994

44-Pin Plastic Quad Flat Pack (B) (MEC) Package 995

SOT311 44-Pin Square Plastic Quad Flat Pack (B) Package 996

64-Pin Plastic Dual In-Line (N) Package 997

68-Pin Plastic Leaded Chip Carrier (A) Package 998

68-Pin Square Ceramic Leaded Chip Carrier with Quartz Window (L) Package 999

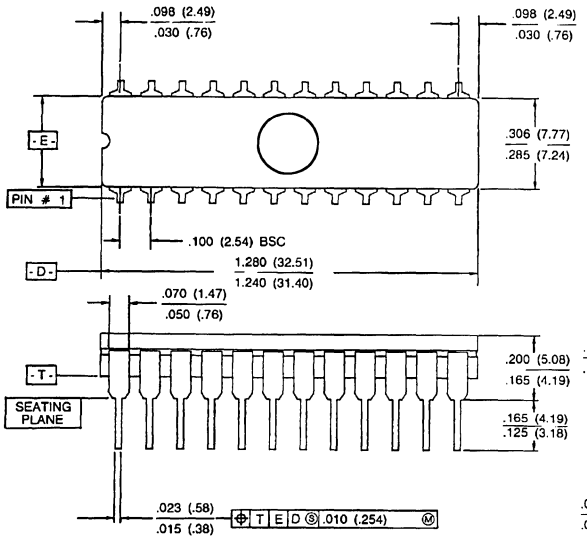
68-Pin Cerquad J-Bend with Quartz Window (K/KA) Package 1000

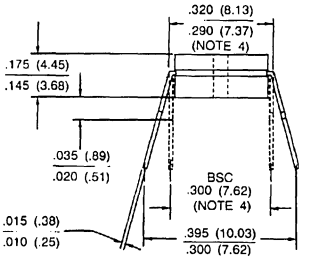
SOT219 80-Pin Plastic Quad Flat Pack (B) Package 1001

SOT318 80-Pin Plastic Quad Flat Pack (B) Package 1002

Package outlines

24-PIN (300 mils wide) CERAMIC DUAL IN-LINE WITH QUARTZ WINDOW (F) PACKAGE



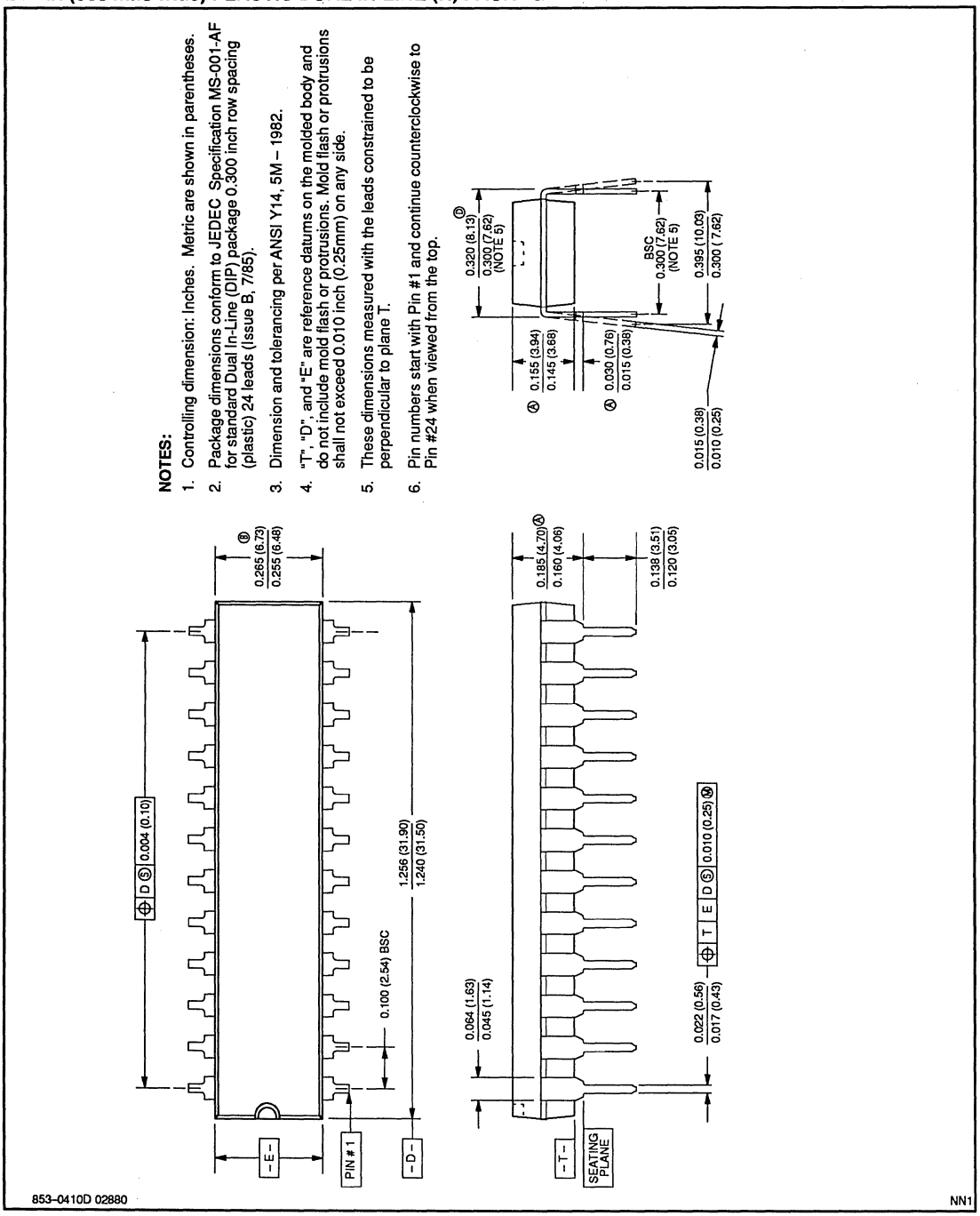


853-0586 84000

NOTES:
1. Controlling dimension: inches. Millimeters are shown in parentheses.
2. Dimensions and tolerancing per ANSI Y14.5M – 1982.
3. "T", "D", and "E" are reference datums on the body and include allowance for glass overrun and meniscus on the seal line, and lid to base mismatch.
4. These dimensions measured with the leads constrained to be perpendicular to plane T.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #20 when viewed from the top.

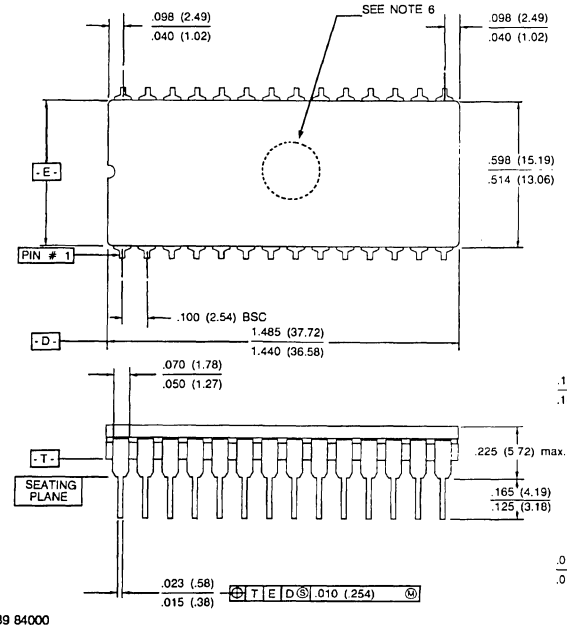
Package outlines

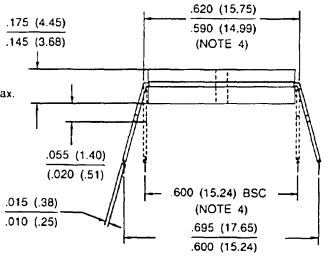
24-PIN (300 mils wide) PLASTIC DUAL IN-LINE (N) PACKAGE



Package outlines

28-PIN (600 mils wide) CERAMIC DUAL IN-LINE WITH QUARTZ WINDOW (F) PACKAGE



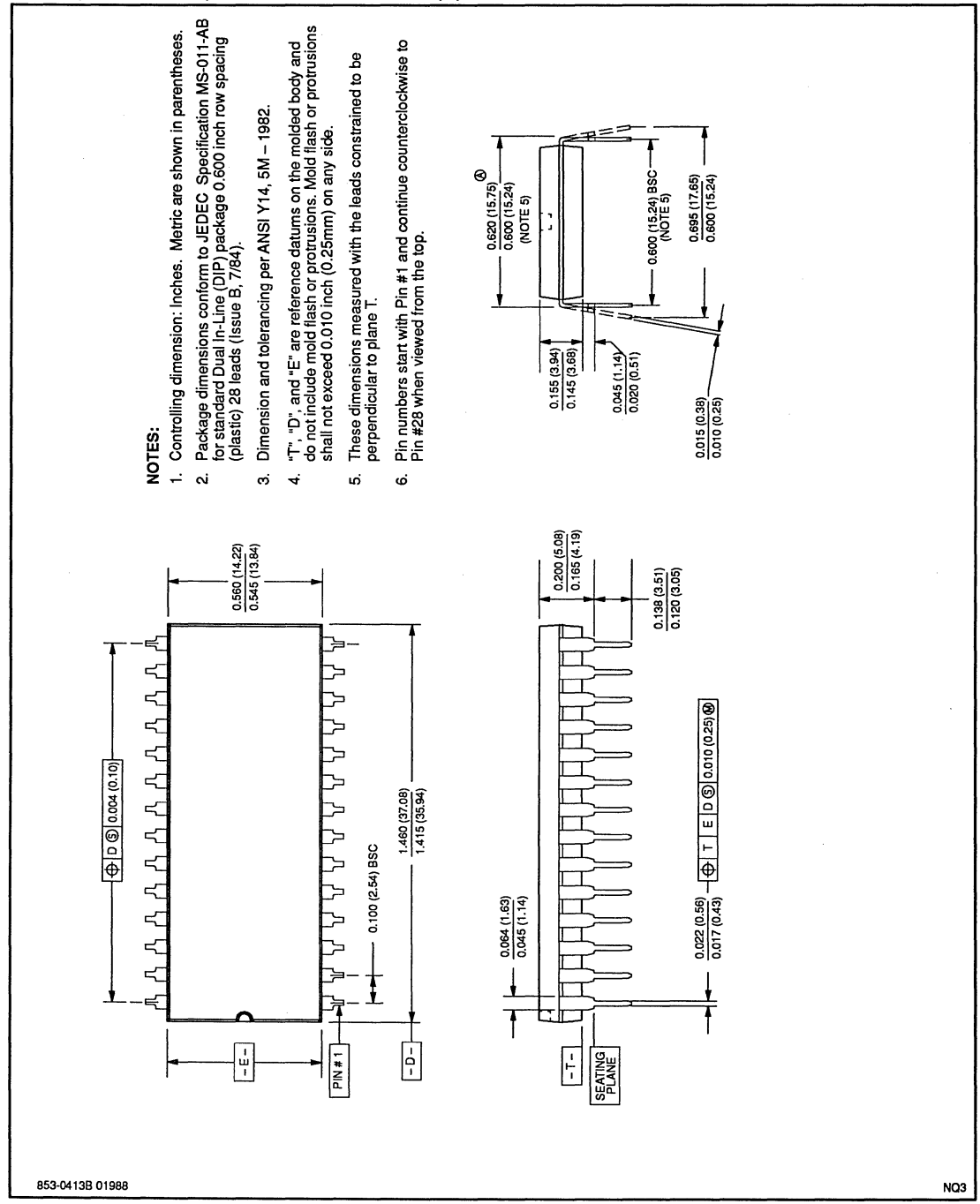


853-0589 84000

NOTES:
1. Controlling dimension: inches. Millimeters are shown in parentheses.
2. Dimensions and tolerancing per ANSI Y14.5M - 1982.
3. "T", "D", and "E" are reference datums on the body and include allowance for glass overrun and meniscus on the seal line, and lid to base mismatch.
4. These dimensions measured with the leads constrained to be perpendicular to plane T.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #28 when viewed from the top.
6. Denotes window location for EPROM products (see FA package).

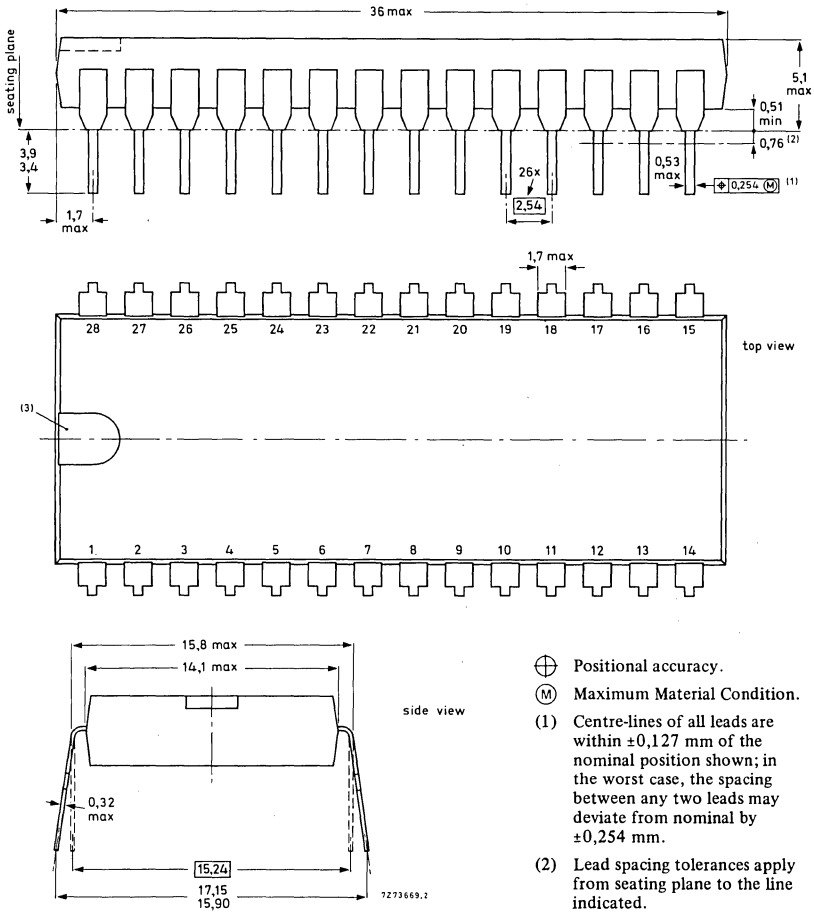
Package outlines

28-PIN (600 mils wide) PLASTIC DUAL IN-LINE (N) PACKAGE



Package outlines

SOT117 28-PIN PLASTIC DUAL IN-LINE (N/P) PACKAGE



⊕ Positional accuracy.

Ⓜ Maximum Material Condition.

(1) Centre-lines of all leads are within ±0,127 mm of the nominal position shown; in the worst case, the spacing between any two leads may deviate from nominal by ±0,254 mm.

(2) Lead spacing tolerances apply from seating plane to the line indicated.

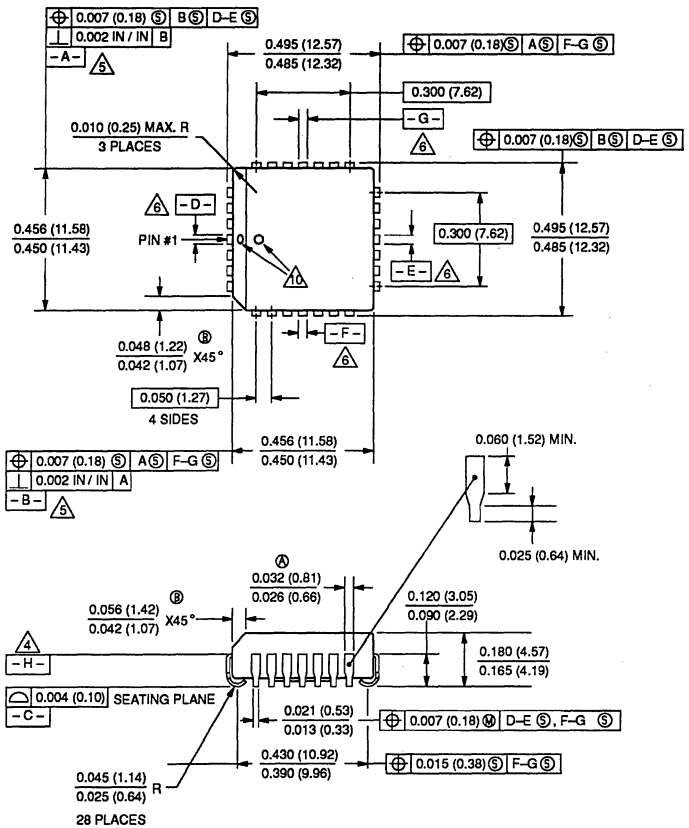
(3) Index may be horizontal as shown, or vertical.

Dimensions in mm

SOT 117 7Z73669.2

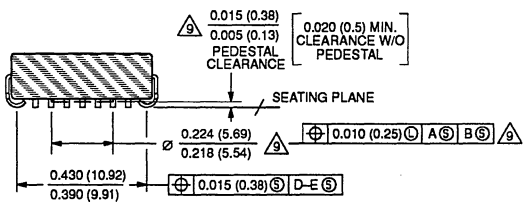
Package outlines

28-PIN PLASTIC LEADED CHIP CARRIER (A) PACKAGE



NOTES

1. Package dimensions conform to JEDEC Specification MO-047-AB for Plastic Leaded Chip Carrier 28 leads, 0.050 inch (1.27mm) lead spacing, square. (Issue A, 10/31/84.)
2. Controlling dimensions: inches. Metric dimensions in mm are shown in parentheses.
3. Dimensioning and tolerancing per ANSI Y14.5M-1982.
4. Datum plane "-H-" located at the top of mold parting line and coincident with top of lead, where lead exits plastic body.
5. Location to datum "-A-" and "-B-" to be determined at plane "-H-". These datums do not include mold flash. Mold flash protrusion shall not exceed 0.010" (0.25mm) on any side.
6. Datum "D–E" and "F–G" are determined where these center leads exit from the body at plane "-H-".
7. Pin numbers continue counterclockwise to Pin 28 (top view).
8. Signetics order code for product packaged in a PLCC is the suffix "A" after the product number.
9. Applicable to packages with pedestal only.
10. Location of Pin #1 mark is optional. Mark on chamfered side is preferred.

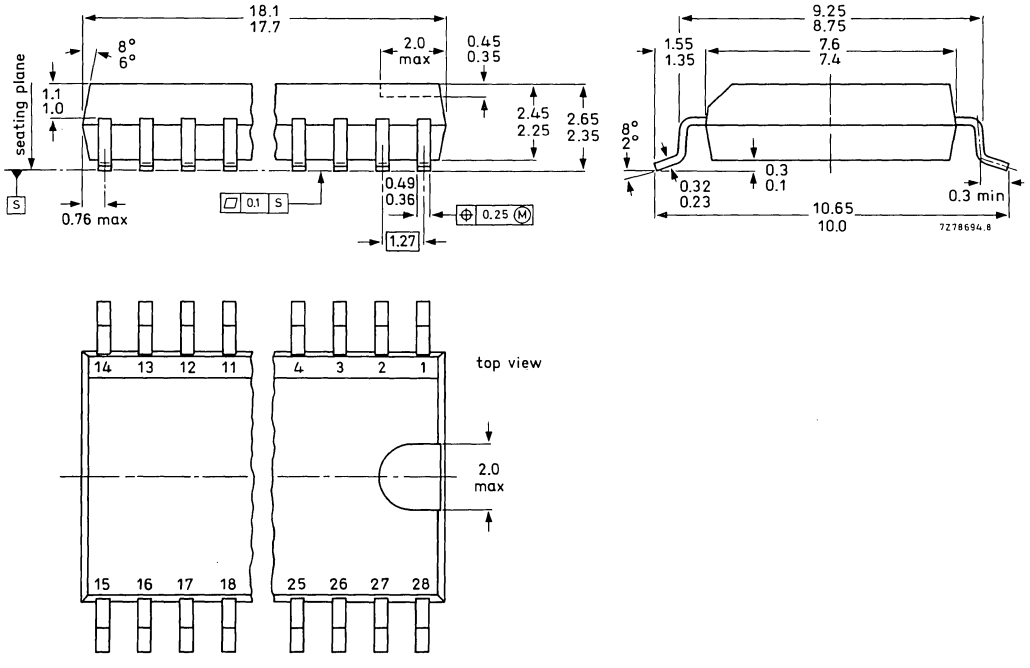


853-0401F 04143

AQ1

Package outlines

SOT136A 28-PIN PLASTIC SO (SMALL OUTLINE) DUAL IN-LINE (D/T)

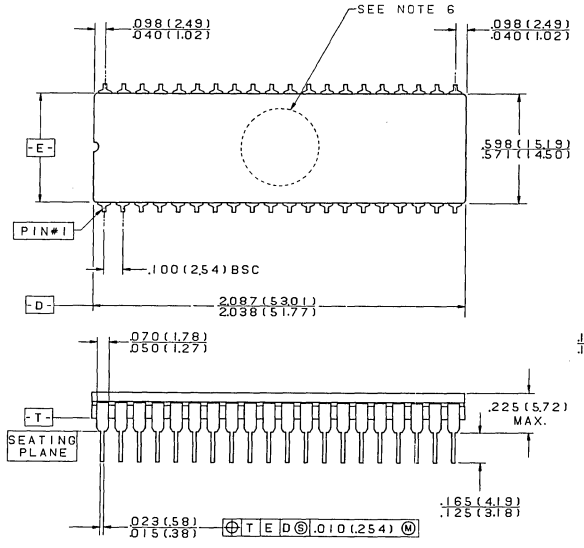


(1) Dimensions in mm.

SOT 136A 7Z78694.8

Package outlines

40-PIN (600 mils wide) CERAMIC DUAL IN-LINE WITH QUARTZ WINDOW (F/FA) PACKAGE



NOTES:

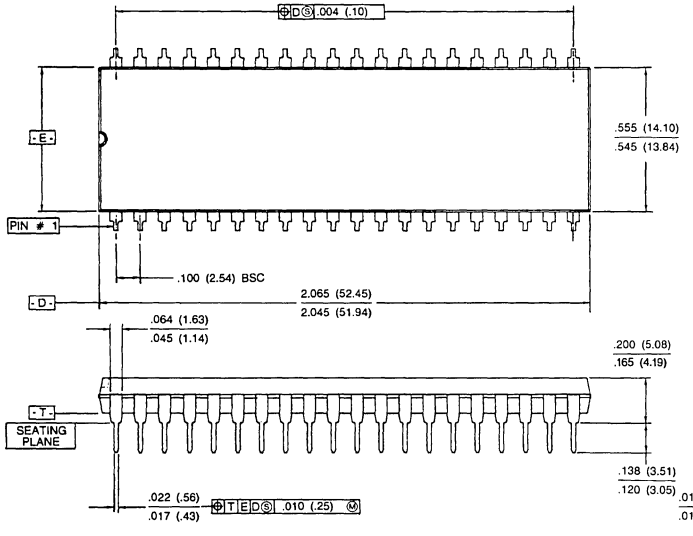
1. CONTROLLING DIMENSION: INCHES. MILLIMETERS ARE SHOWN IN PARENTHESIS.
2. DIMENSION AND TOLERANCING PER ANSI Y14.5M-1982.
3. "T", "D", AND "E" ARE REFERANCE DATUMS ON THE BODY AND INCLUDE ALLOWANCE FOR GLASS OVERRUN AND MENISCUS ON THE SEAL LINE, AND LID TO BASE MISMATCH.
4. THESE DIMENSIONS MEASURED WITH THE LEADS CONSTRAINED TO BE PERPENDICULAR TO PLANE T.
5. PIN NUMBERS START WITH PIN#1 AND CONTINUE COUNTERCLOCKWISE TO PIN#40 WHEN VIEWED FROM THE TOP.
6. DENOTES WINDOW LOCATION FOR EPROM PRODUCTS.

853-0590 95528

FW3

Package outlines

40-PIN (600 mils wide) PLASTIC DUAL IN-LINE (N) PACKAGE



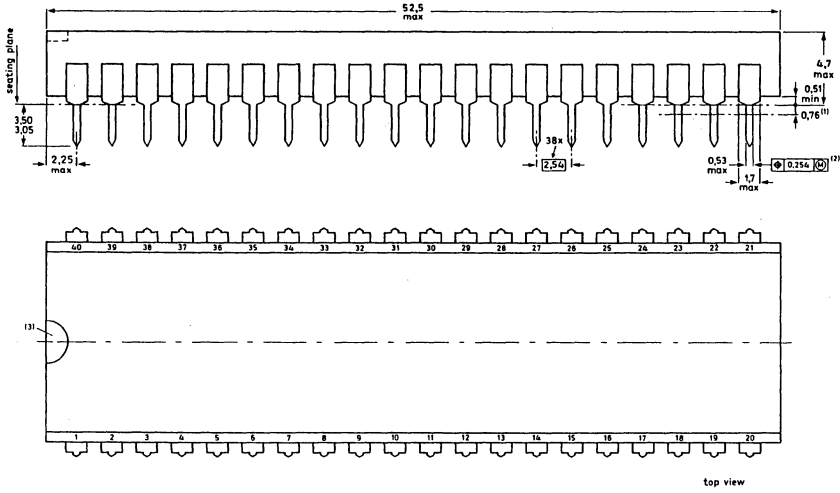
NOTES:

1. Controlling dimension: inches. Metric are shown in parentheses.
2. Package dimensions conform to JEDEC specification MS-011-AC for standard dual in-line (DIP) package .600 inch row spacing (PLASTIC) 40 leads (issue B. 7/85)
3. Dimensions and tolerancing per ANSI Y14. 5M-1982.
4. "T", "D" and "E" are reference datums on the molded body and do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010 inch (.25mm) on any side.
5. These dimensions measured with the leads constrained to be perpendicular to plane T.
6. Pin numbers start with pin #1 and continue counterclockwise to pin #40 when viewed from the top.

853-0415 84971

Package outlines

SOT129 40-PIN PLASTIC DUAL IN-LINE (P/N) PACKAGE



(1) Centre-lines of all leads are within ±0.127 mm of the nominal position shown; in the worst case, the spacing between any two leads may deviate from nominal by ±0.254 mm.

(2) Lead spacing tolerances apply from seating plane to the line indicated.

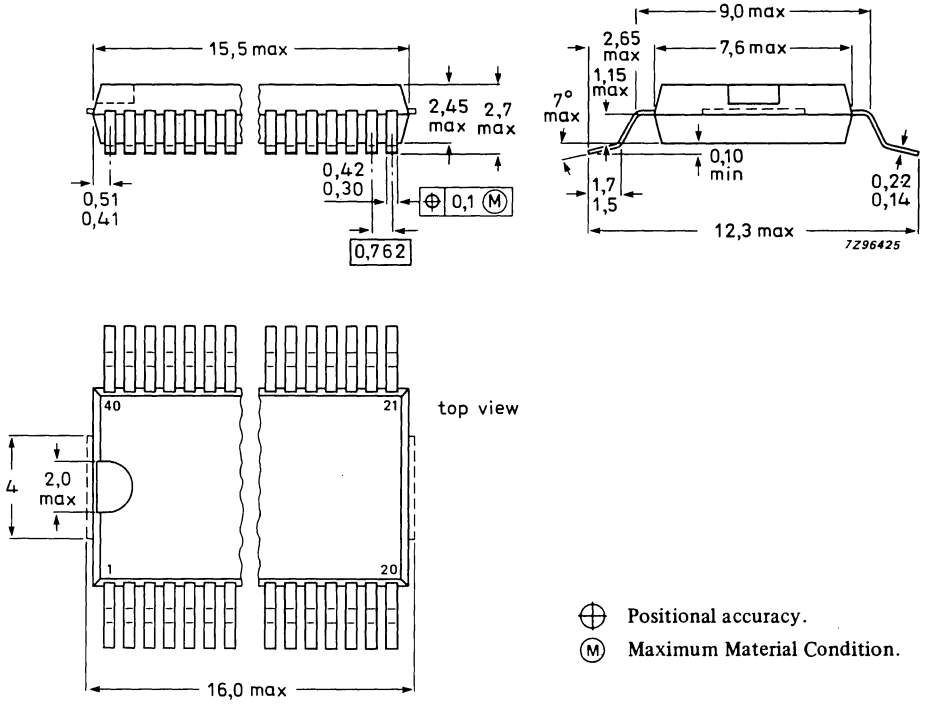
(3) Index may be horizontal as shown, or vertical

(4) Dimensions in mm.

SOT 129 7Z70128.5

Package outlines

SOT158A 40-PIN PLASTIC VSO (VERY SMALL OUTLINE) DUAL IN-LINE (D/T) PACKAGE

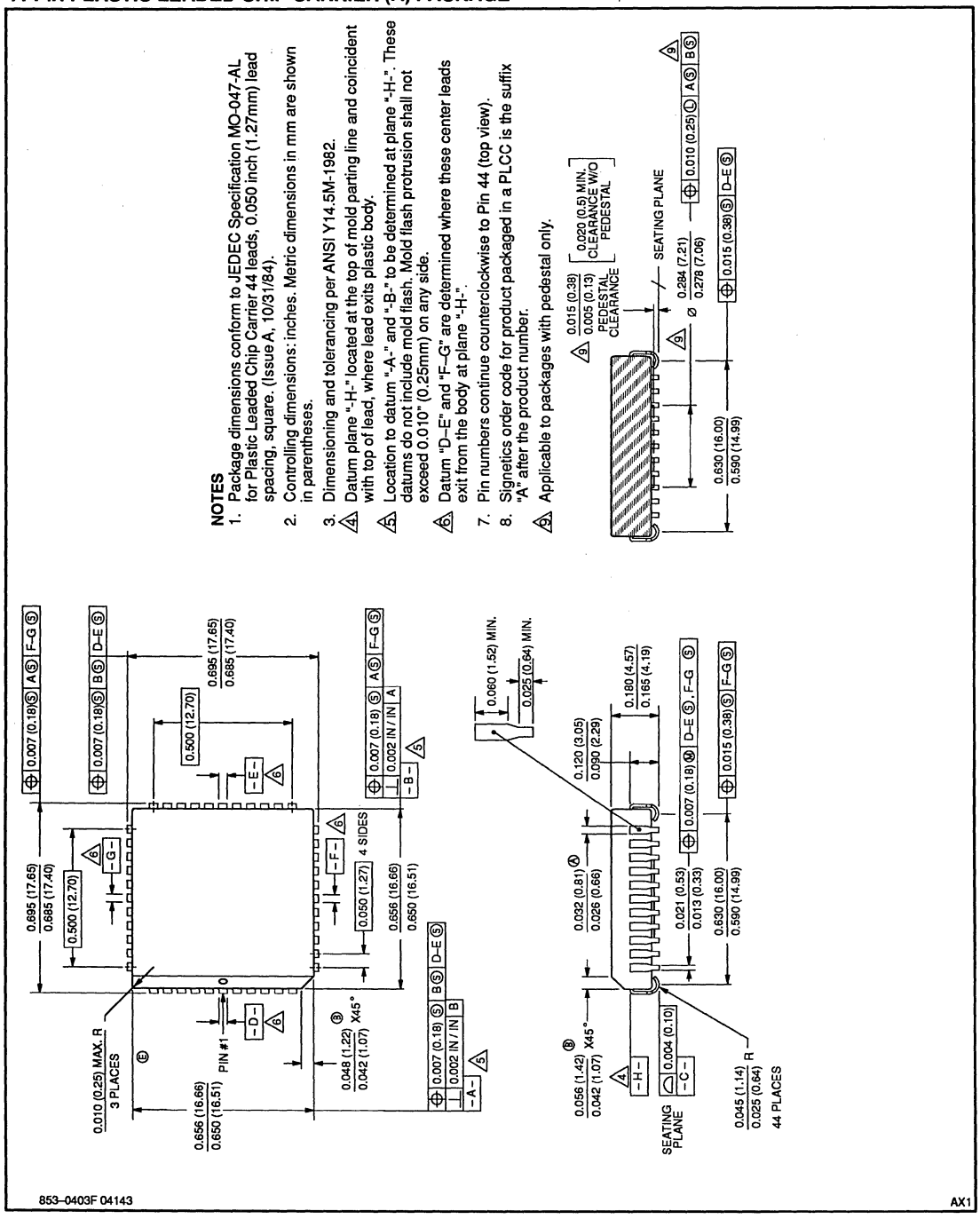


Dimensions in mm

SOT 158A 7Z96425

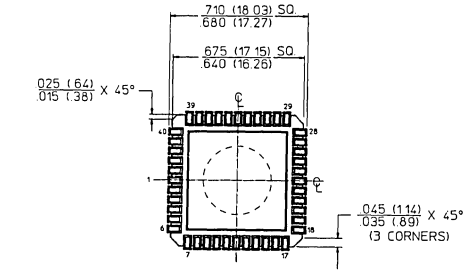
Package outlines

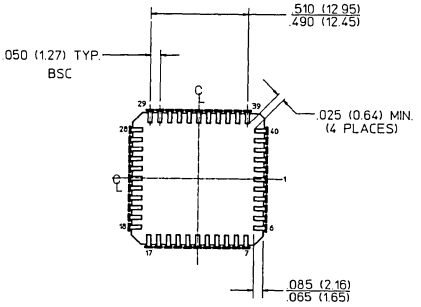
44-PIN PLASTIC LEADED CHIP CARRIER (A) PACKAGE

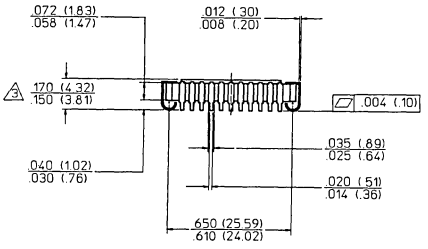


Package outlines

44-PIN SQUARE CERAMIC LEADED CHIP CARRIER WITH QUARTZ WINDOW (L) PACKAGE







NOTES:

1. CONTROLLING DIMENSIONS: INCHES. MILLIMETERS ARE SHOWN IN PARENTHESIS.
2. DIMENSION AND TOLERANCING PER ANSI Y14.5M-1982.
3. WHEN A WINDOW LID IS USED, THE OVERALL PACKAGE THICKNESS MUST INCREASE BY A MINIMUM OF .010 INCH (0.254mm) AND A MAXIMUM OF .040 INCH (1.020mm).

853-1268 95007

Package outlines

44-PIN CERQUAD J-BEND WITH QUARTZ WINDOW (K/KA) PACKAGE

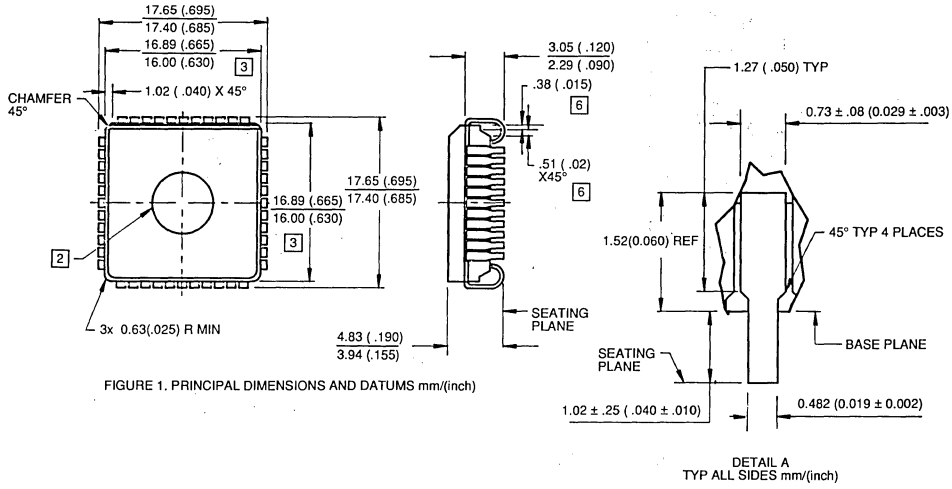


FIGURE 1. PRINCIPAL DIMENSIONS AND DATUMS mm/(inch)

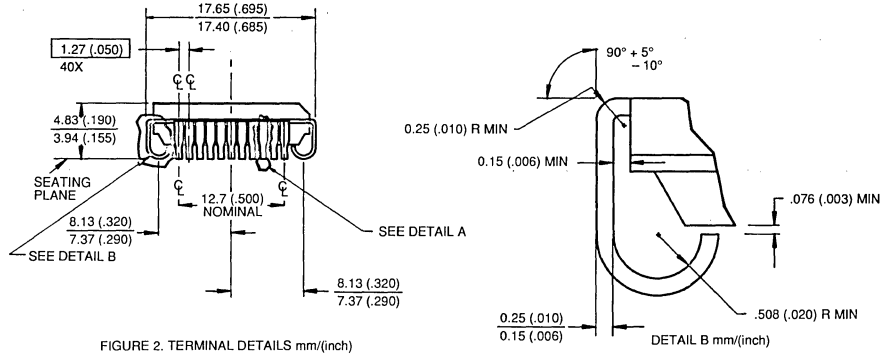


FIGURE 2. TERMINAL DETAILS mm/(inch)

1. ALL DIMENSIONS AND TOLERANCES TO CONFORM TO ANSI Y14.5-1982.
2. UV WINDOW IS OPTIONAL.
3. DIMENSIONS DO NOT INCLUDE GLASS PROTRUSION. GLASS PROTRUSION TO BE 0.005 INCHES MAX ON EACH SIDE.
4. CONTROLLING DIMENSION MILLIMETERS.
5. ALL DIMENSIONS AND TOLERANCES INCLUDE LEAD TRIM OFFSET AND LEAD PLATING FINISH.
6. BACKSIDE SOLDER RELIEF IS OPTIONAL AND DIMENSIONS ARE FOR REFERENCE ONLY.

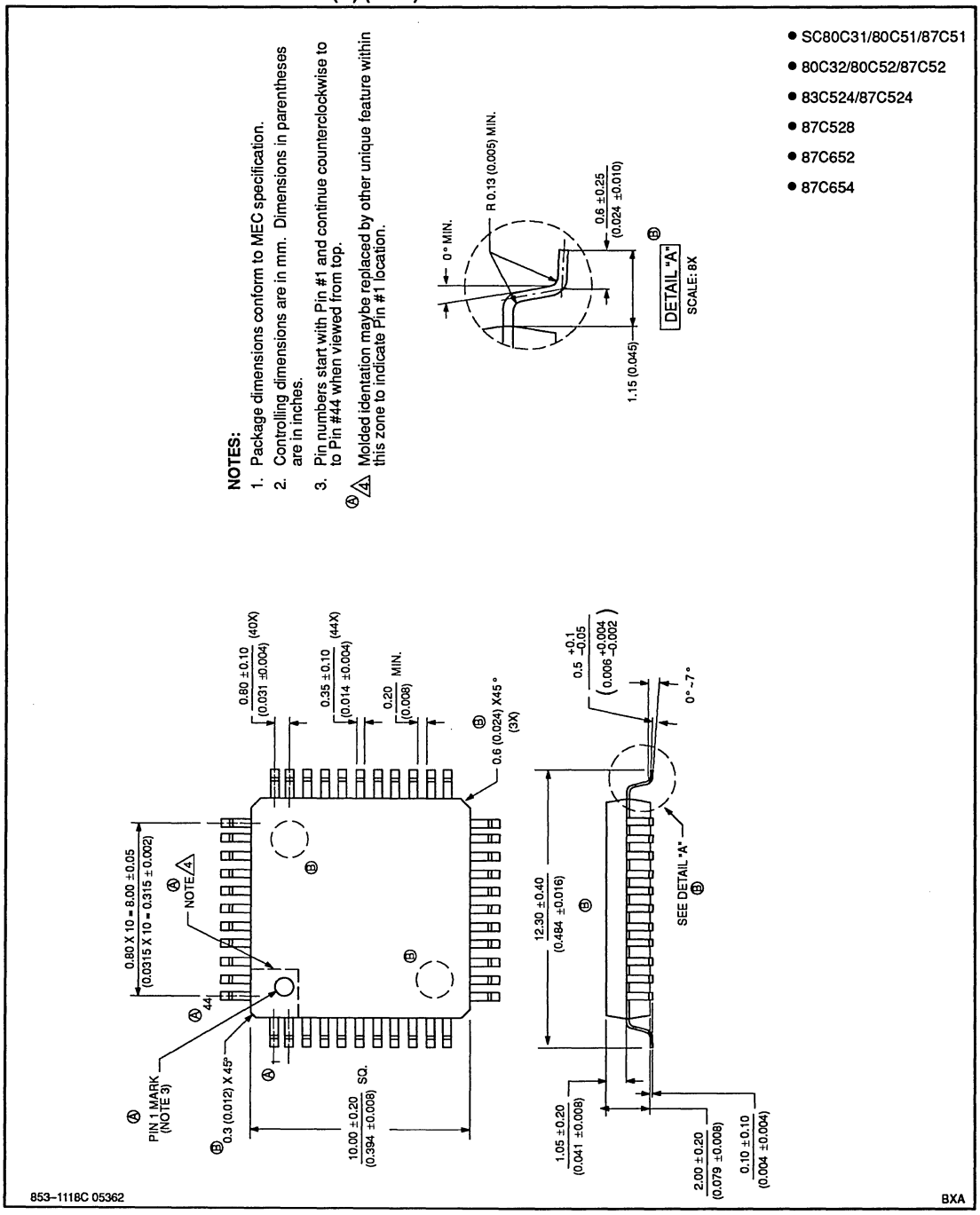
853-1472 00440

KB1

Package outlines

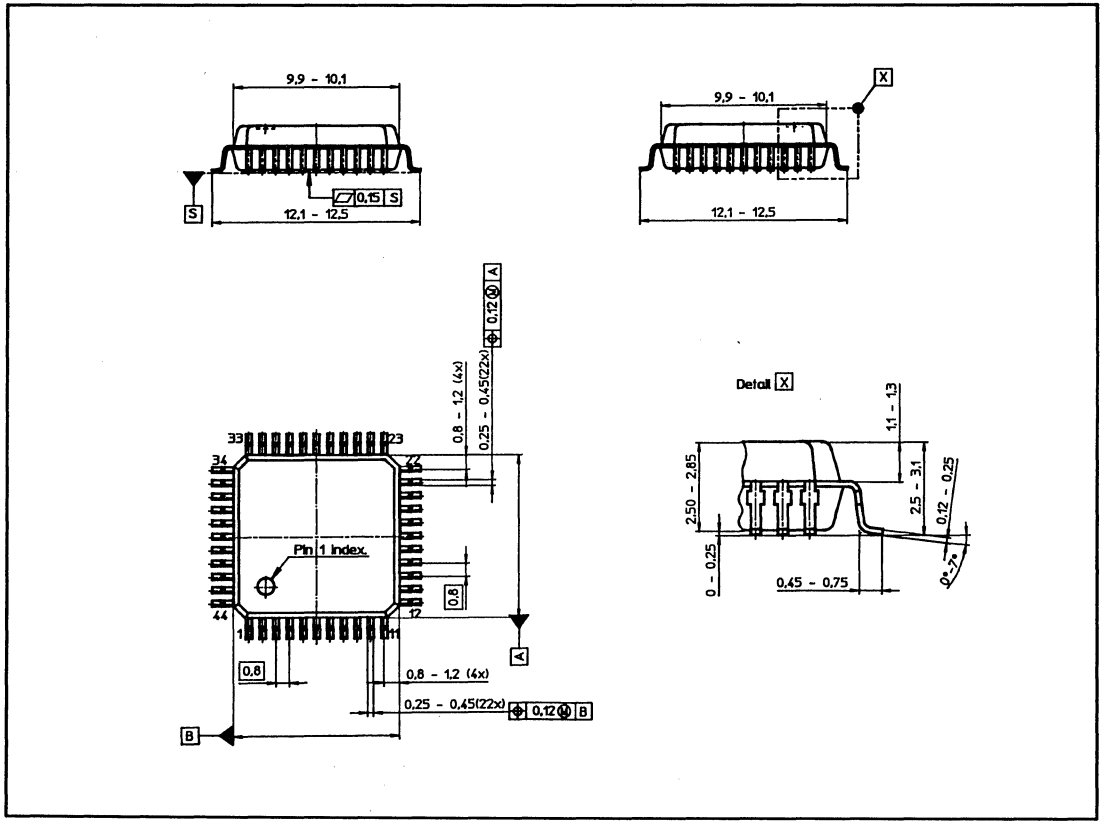
44-PIN PLASTIC QUAD FLAT PACK (B) (MEC) PACKAGE

- SC80C31/80C51/87C51
- 80C32/80C52/87C52
- 83C524/87C524
- 87C528
- 87C652
- 87C654



Package outlines

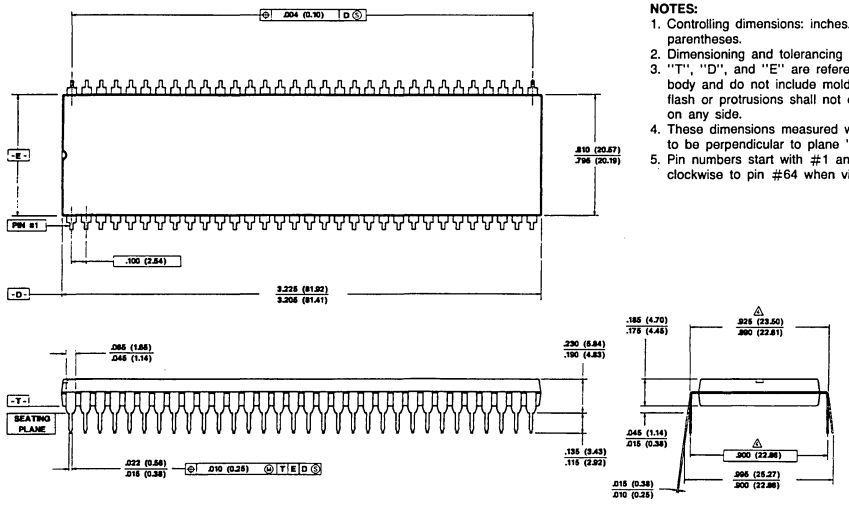
SOT311 44-PIN SQUARE PLASTIC QUAD FLAT PACK (B) PACKAGE



- PCB/PCF/PCA 80C31/80C51
- 80C528/83C528
- 80C652/83C652
- 80C654/83C654
- 80CE654/83CE654
- 80C851/83C851

Package outlines

64-PIN PLASTIC DUAL IN-LINE (N) PACKAGE



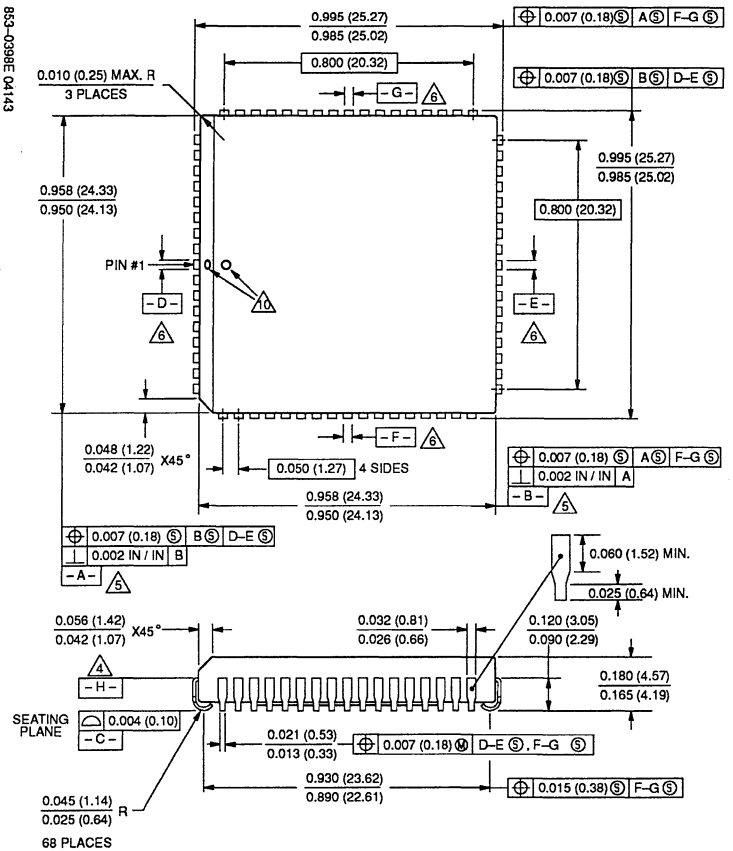
NOTES:
1. Controlling dimensions: inches. Metric are shown in parentheses.
2. Dimensioning and tolerancing per ANSI Y14.5M-1982.
3. "T", "D", and "E" are reference Datums on the molded body and do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010 inch (25mm) on any side.
4. These dimensions measured with the leads constrained to be perpendicular to plane "T".
5. Pin numbers start with #1 and continue counter-clockwise to pin #64 when viewed from the top.

853-0414 93945

NS4

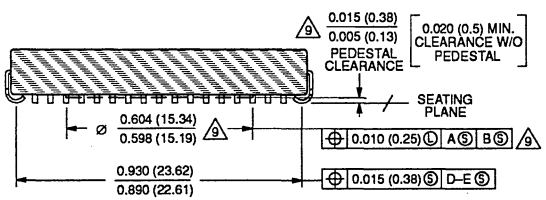
Package outlines

68-PIN PLASTIC LEADED CHIP CARRIER (A) PACKAGE



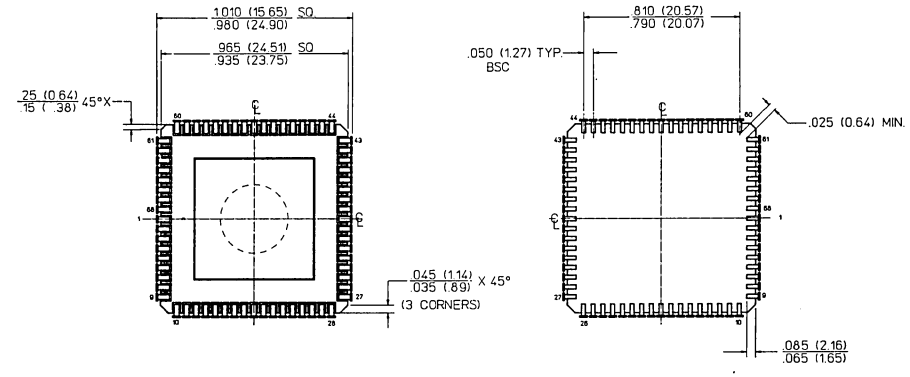
NOTES

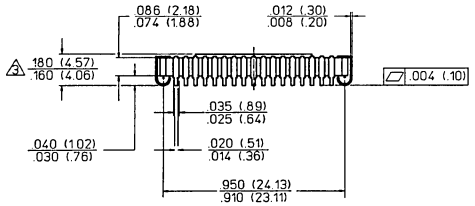
1. Package dimensions conform to JEDEC Specification MO-047-AE for Plastic Leaded Chip Carrier 68 leads, 0.050 inch lead spacing, square. (Issue A, 10/31/84).
2. Controlling dimensions: inches. Metric dimensions in mm are shown in parentheses.
3. Dimensioning and tolerancing per ANSI Y14.5M-1982.
4. Datum plane "-H-" located at the top of mold parting line and coincident with top of lead, where lead exits plastic body.
5. Location to datum "-A-" and "-B-" to be determined at plane "-H-". These datums do not include mold flash. Mold flash protrusion shall not exceed 0.010" (0.25mm) on any side.
6. Datum "D–E" and "F–G" are determined where these center leads exit from the body at plane "-H-".
7. Pin numbers continue counterclockwise to Pin 68 (top view).
8. Signetics order code for product packaged in a PLCC is the suffix "A" after the product number.
9. Applicable to packages with pedestal only.
10. Location of Pin #1 mark is optional. Mark on chamfered side is preferred.



Package outlines

68-PIN SQUARE CERAMIC LEADED CHIP CARRIER WITH QUARTZ WINDOW (L) PACKAGE





NOTES:

1. CONTROLLING DIMENSIONS: INCHES. MILLIMETERS ARE SHOWN IN PARENTHESIS.
2. DIMENSION AND TOLERANCING PER ANSI Y14.5M-1982.
3. WHEN A WINDOW LID IS USED, THE OVERALL PACKAGE THICKNESS MUST INCREASED BY A MINIMUM OF .010 INCH (0.25mm) AND A MAXIMUM OF .040 INCH (1.020mm).

853-1240 95007

Package outlines

68-PIN CERQUAD J-BEND WITH QUARTZ WINDOW (K/KA) PACKAGE

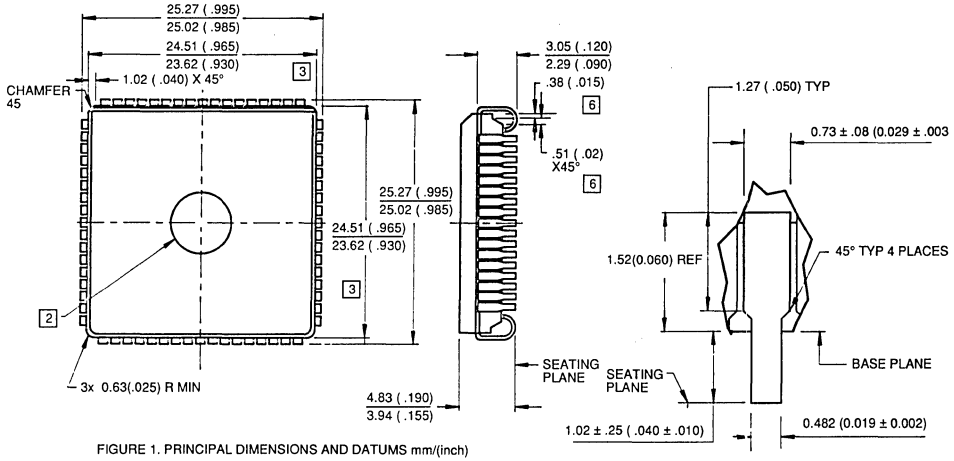


FIGURE 1. PRINCIPAL DIMENSIONS AND DATUMS mm/(inch)

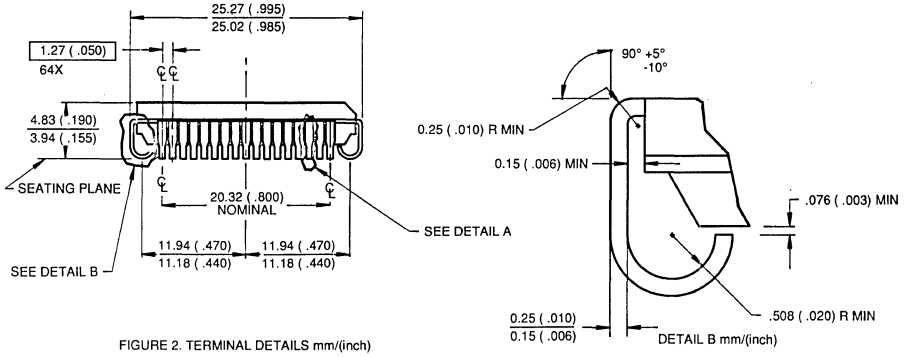


FIGURE 2. TERMINAL DETAILS mm/(inch)

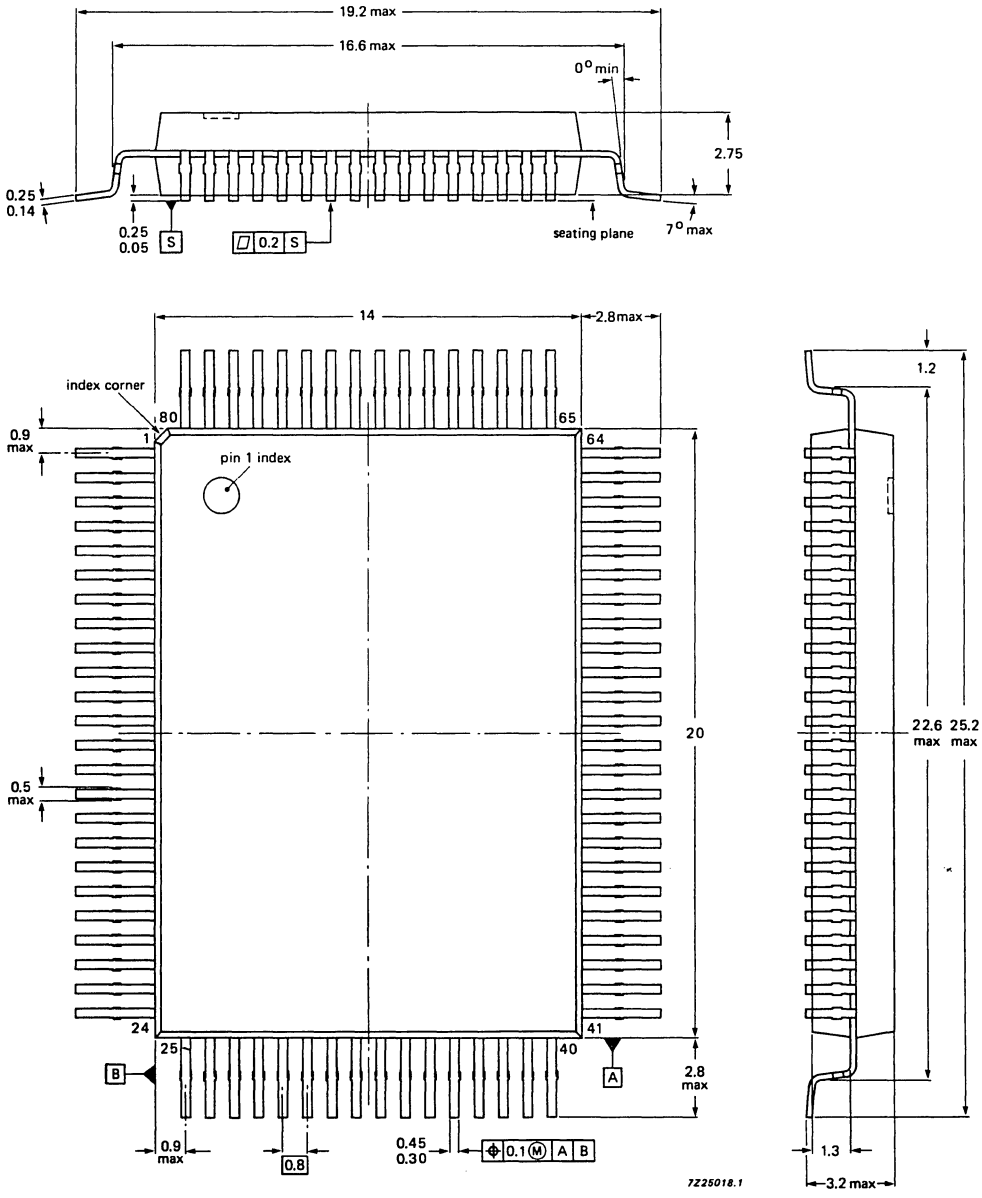
1. ALL DIMENSIONS AND TOLERANCES TO CONFORM TO ANSI Y14.5-1982.
2. UV WINDOW IS OPTIONAL.
3. DIMENSIONS DO NOT INCLUDE GLASS PROTRUSION. GLASS PROTRUSION TO BE 0.005 INCHES MAX ON EACH SIDE.
4. CONTROLLING DIMENSION MILLIMETERS.
5. ALL DIMENSIONS AND TOLERANCES INCLUDE LEAD TRIM OFFSET AND LEAD PLATING FINISH.
6. BACKSIDE SOLDER RELIEF IS OPTIONAL AND DIMENSIONS ARE FOR REFERENCE ONLY.

853-1473 00441

KC1

Package outlines

SOT219 80-PIN PLASTIC QUAD FLAT PACK (B) PACKAGE



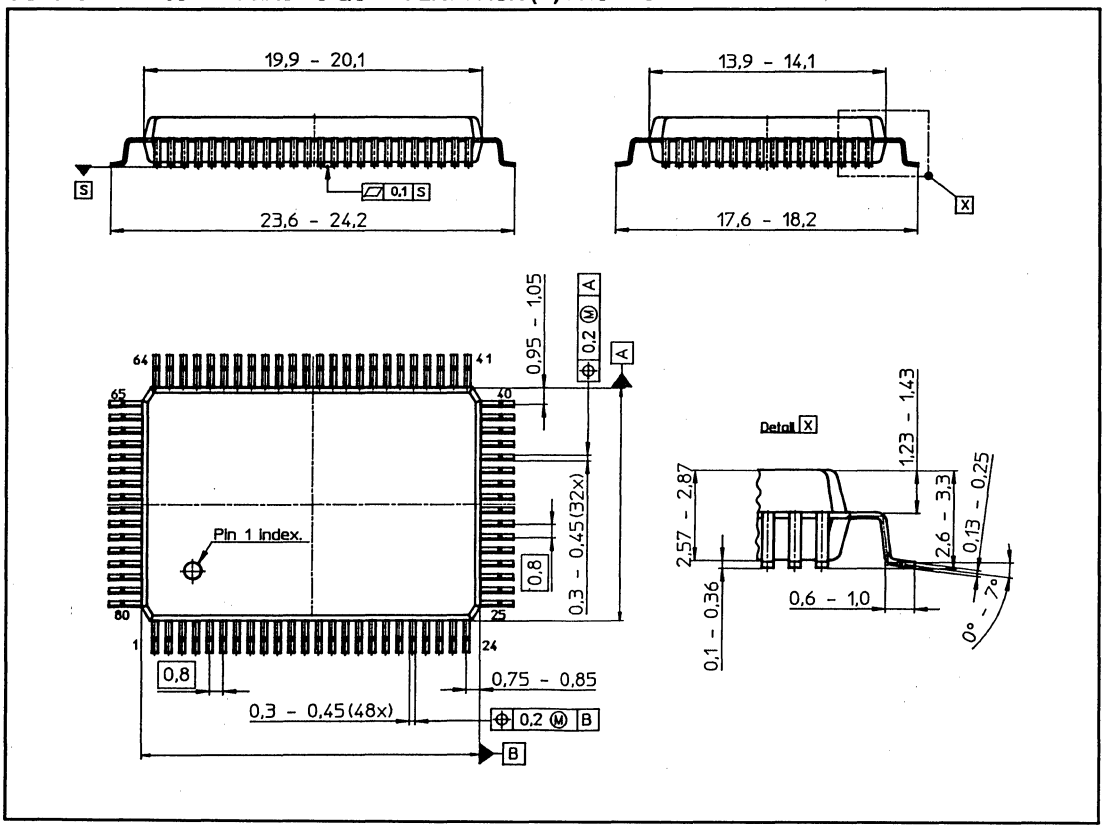
⊕ Positional accuracy.

Ⓜ Maximum Material Condition.

Dimensions in mm

Package outlines

SOT318 80-PIN PLASTIC QUAD FLAT PACK (B) PACKAGE



Section 9
Sales Offices, Representatives & Distributors

Sales Offices, Representatives and Distributors

SIGNETICS HEADQUARTERS
811 East Arques Avenue
P.O. Box 3409
Sunnyvale, CA 94088-3409

ALABAMA
Huntsville
Phone: (205) 464-0111

CALIFORNIA
Calabasas
Phone: (818) 880-6304

Irvine
Phone: (714) 833-8980
(714) 752-2780

San Diego
Phone: (619) 560-0242

Sunnyvale
Phone: (408) 991-3737

COLORADO
Englewood
Phone: (303) 792-9011

GEORGIA
Atlanta
Phone: (404) 594-1392

ILLINOIS
Itasca
Phone: (708) 250-0050

INDIANA
Kokomo
Phone: (317) 459-5355

MASSACHUSETTS
Westford
Phone: (508) 692-6211

MICHIGAN
Novi
Phone: (313) 347-1400

NEW JERSEY
Toms River
Phone: (908) 505-1200

NEW YORK
Wappingers Falls
Phone: (914) 297-4074

OHIO
Columbus
Phone: (614) 888-7143

OREGON
Beaverton
Phone: (503) 627-0110

PENNSYLVANIA
Plymouth Meeting
Phone: (215) 825-4404

TENNESSEE
Greeneville
Phone: (615) 639-0251

TEXAS
Austin
Phone: (512) 339-9945

Richardson
Phone: (214) 644-1610

CANADA
SIGNETICS CANADA, LTD.
Etobicoke, Ontario
Phone: (416) 626-6676

Nepean, Ontario
Phone: (613) 225-5467

REPRESENTATIVES

ALABAMA
Huntsville
Elcom, Inc.
Phone: (205) 830-4001

ARIZONA
Scottsdale
Thom Luke Sales, Inc.
Phone: (602) 941-1901

CALIFORNIA
Orangevale
Webster Associates
Phone: (916) 989-0843

San Jose
B.A.E. Sales, Inc.
Phone: (408) 452-8133

COLORADO
Englewood
Thom Luke Sales, Inc.
Phone: (303) 649-9717

CONNECTICUT
Wallingford
JEBCO
Phone: (203) 265-1318

FLORIDA
Oviedo
Conley and Assoc., Inc.
Phone: (407) 365-3283

GEORGIA
Norcross
Elcom, Inc.
Phone: (404) 447-8200

ILLINOIS
Hoffman Estates
Micro-Tex, Inc.
Phone: (708) 765-3000

INDIANA
Indianapolis
Mohrfield Marketing, Inc.
Phone: (317) 546-6969

IOWA
Cedar Rapids
J.R. Sales
Phone: (319) 393-2232

MARYLAND
Columbia
Third Wave Solutions, Inc.
Phone: (301) 290-5990

MASSACHUSETTS
Chelmsford
JEBCO
Phone: (508) 256-5800

MICHIGAN
Brighton
AP Associates
Phone: (313) 229–6550

MINNESOTA
Eden Prairie
High Technology Sales
Phone: (612) 944-7274

MISSOURI
Bridgeton
Centech, Inc.
Phone: (314) 291-4230

Raytown
Centech, Inc.
Phone: (816) 358-8100

NEW YORK
Ithaca
Bob Dean, Inc.
Phone: (607) 257-1111

Rockville Centre
S-J Associates
Phone: (516) 536-4242

Wappingers Falls
Bob Dean, Inc.
Phone: (914) 297-6406

NORTH CAROLINA
Matthews
ADI, Inc.
Phone: (704) 847-4323

Smithfield
ADI, Inc.
Phone: (919) 934-8136

OHIO
Aurora
InterActive Technical Sales, Inc.
Phone: (216) 562-2050

Dayton
InterActive Technical Sales, Inc.
Phone: (513) 436-2230

Dublin
InterActive Technical Sales, Inc.
Phone: (614) 792-5900

OREGON
Beaverton
Western Technical Sales
Phone: (503) 644-8860

PENNSYLVANIA
Hatboro
Delta Technical Sales, Inc.
Phone: (215) 957-0600

TEXAS
Austin
Synergistic Sales, Inc.
Phone: (512) 346-2122

Houston
Synergistic Sales, Inc.
Phone: (713) 937-1990

Richardson
Synergistic Sales, Inc.
Phone: (214) 644-3500

UTAH
Salt Lake City
Electrodyne
Phone: (801) 264-8050

WASHINGTON
Bellevue
Western Technical Sales
Phone: (206) 641-3900

Spokane
Western Technical Sales
Phone: (509) 922-7600

WISCONSIN
Waukesha
Micro-Tex, Inc.
Phone: (414) 542-5352

CANADA
Calgary, Alberta
Tech-Trek, Ltd.
Phone: (403) 241-1719

Mississauga, Ontario
Tech-Trek, Ltd.
Phone: (416) 238-0366

Nepean, Ontario
Tech-Trek, Ltd.
Phone: (613) 225-5161

Richmond, B.C.
Tech-Trek, Ltd.
Phone: (604) 276-8735

Ville St. Laurent, Quebec
Tech-Trek, Ltd.
Phone: (514) 337-7540

PUERTO RICO
Santurce
Mectron Group
Phone: (809) 723-6165

DISTRIBUTORS

Contact one of our local distributors:
Almac/Arrow Electronics
Anthem Electronics
Arrow/Schweber Electronics
Falcon Electronics, Inc.
Gerber Electronics
Hamilton/Avnet Electronics
Marshall Industries
Wyle/EMG
Zentronics, Ltd.

01/06/92

Data handbook system — Appendix A

INTRODUCTION

Our data handbook system comprises more than 65 books with subjects including electronic components, subassemblies and magnetic products. The handbooks are classified into seven series:

INTEGRATED CIRCUITS;
DISCRETE SEMICONDUCTORS;
DISPLAY COMPONENTS;
PASSIVE COMPONENTS;
PROFESSIONAL COMPONENTS;
MAGNETIC PRODUCTS;
LIQUID CRYSTAL DISPLAYS.

Data handbooks contain all pertinent data available at the time of publication and each is revised and reissued regularly.

Loose data sheets are sent to subscribers to keep them up-to-date on additions or alterations made during the lifetime of a data handbook.

Catalogs are available for selected product ranges (some catalogs are also on floppy discs).

For more information about data handbooks, catalogs and subscriptions, contact one of the organizations listed on the back cover of this handbook. Product specialists are at your service and inquiries are answered promptly.

INTEGRATED CIRCUITS

IC01	Radio, Audio and Associated Systems Bipolar, MOS
IC02a/b	Video and Associated Systems Bipolar, MOS
IC03	ICs for Telecom Subscriber Sets, Cordless, Mobile and Cellular Telephones, Radio Pagers
IC04	HE4000B Logic Family CMOS
IC05	Advanced Low-power Schottky (ALS) Logic Series
IC06	High-speed CMOS; 74HC/HCT/HCU Logic Family
IC07	Advanced CMOS Logic (ACL)
IC08	10/100k ECL Logic/Memory/PLD
IC09	TTL Logic Series

INTEGRATED CIRCUITS (continued)

IC10	Memories MOS, TTL, ECL
IC11	Linear Products
IC12	I^2C-bus-compatible ICs
IC13	Programmable Logic Devices
IC14	8048-Based 8-Bit Microcontrollers
IC15	FAST TTL Logic Series
IC15 supplement: Additional FAST Data	
IC16	CMOS Integrated Circuits for Clocks and Watches
IC17	ICs for Telecom ISDN
IC18	Microprocessors and Peripherals
IC19	Data Communication Products
IC20	80C51-Based 8-Bit Microcontrollers
IC23	Advanced BiCMOS Interface Logic

DISCRETE SEMICONDUCTORS

SC01	Diodes
SC02	Power Diodes
SC03	Thyristors and Triacs
SC04	Small Signal Transistors
SC05	Low-frequency Power Transistors and Hybrid IC Power Modules
SC06	High-voltage and Switching Power Transistors
SC07	Small-signal Field-effect Transistors
SC08a	RF Power Bipolar Transistors
SC08b	RF Power MOS Transistors
SC09	RF Power Modules
SC10	Surface Mounted Semiconductors
SC12	Optocouplers
SC13	Power MOS Transistors
SC14	Wideband Transistors and Wideband Hybrid IC Modules
SC15	Microwave Transistors
SC17	Semiconductor Sensors

Data handbook system — Appendix A

DISPLAY COMPONENTS

DC01	Colour Display Components Colour TV Picture Tubes and Assemblies Color Monitor Tube Assemblies
DC02	Monochrome Monitor Tubes and Deflection Units
DC03	Television Tuners, Coaxial Aerial Input Assemblies
DC04	Loudspeakers
DC05	Flyback Transformers, Mains Transformers and General-purpose FXC Assemblies

PASSIVE COMPONENTS

PA01	Electrolytic Capacitors; Solid and Non-solid
PA02	Varistors, Thermistors and Sensors
PA03	Potentiometers and Switches
PA04	Variable Capacitors
PA05	Film Capacitors
PA06	Ceramic Capacitors
PA07	Piezoelectric Quartz Devices
PA08	Fixed Resistors
PA11	Quartz Oscillators

PROFESSIONAL COMPONENTS

PC01	High-power Klystrons and Accessories
PC02	Cathode-ray Tubes
PC03	Geiger-Muller Tubes
PC04	Photo Multipliers
PC05	Plumbicon Camera Tubes and Accessories
PC06	Circulators and Isolators
PC07	Vidicon and Newvicon Camera Tubes and Deflection Units
PC08	Image Intensifiers
PC09	Dry-reed Switches
PC11	Solid-state Image Sensors and Peripheral Integrated Circuits
PC12	Electron Multipliers

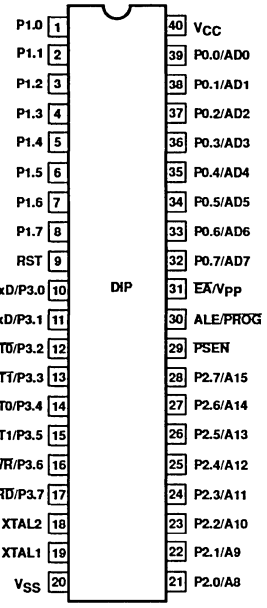
MAGNETIC PRODUCTS

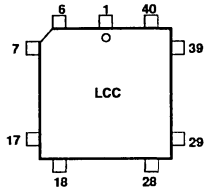
MA01	Soft Ferrites
MA02	Permanent Magnet Materials
MA03	Piezoelectric Ceramics

LIQUID CRYSTAL DISPLAYS

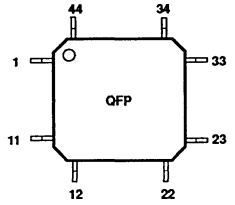
LCD01	Liquid Crystal Displays and Driver ICs for LCDs

8031AH/8051AH, 80C31/80C51/87C51, 80C652/83C652/87C652*, 83C654/87C654*, 80C851/83C851





Pin	Function	Pin	Function
1	NC	23	NC
2	P1.0	24	P2.0/A8
3	P1.1	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6	30	P2.6/A14
9	P1.7	31	P2.7/A15
10	RST	32	PSEN
11	P3.0/RxD	33	ALE/PROG
12	NC	34	NC
13	P3.1/TxD	35	EA/VPP
14	P3.2/INT0	36	P0.7/AD7
15	P3.3/INT1	37	P0.6/AD6
16	P3.4/T0	38	P0.5/AD5
17	P3.5/T1	39	P0.4/AD4
18	P3.6/WR	40	P0.3/AD3
19	P3.7/RD	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	VSS	44	VCC



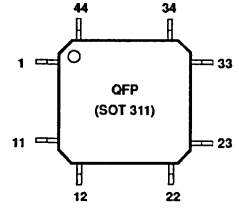
Pin	Function	Pin	Function
1	P1.5	23	P2.5/A13
2	P1.6	24	P2.6/A14
3	P1.7	25	P2.7/A15
4	RST	26	PSEN
5	P3.0/RxD	27	ALE/PROG
6	NC	28	NC
7	P3.1/TxD	29	EA/VPP
8	P3.2/INT0	30	P0.7/AD7
9	P3.3/INT1	31	P0.6/AD6
10	P3.4/T0	32	P0.5/AD5
11	P3.5/T1	33	P0.4/AD4
12	P3.6/WR	34	P0.3/AD3
13	P3.7/RD	35	P0.2/AD2
14	XTAL2	36	P0.1/AD1
15	XTAL1	37	P0.0/AD0
16	VSS	38	VCC
17	NC	39	NC
18	P2.0/A8	40	P1.0
19	P2.1/A9	41	P1.1
20	P2.2/A10	42	P1.2
21	P2.3/A11	43	P1.3
22	P2.4/A12	44	P1.4

* P1.6 and P1.7 have the alternate functions SCL and SDA, respectively, on the 80C652/83C652/87C652 and 83C654/87C654.

NOTES TO QFP ONLY:

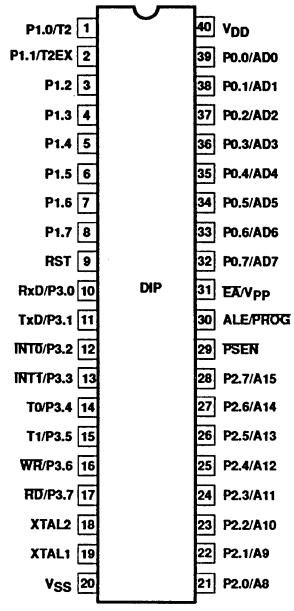
1. Due to EMC improvements, it is advised to connect pins 6, 28, 39 to V_{SS} on the 80C652/83C652, and 83C654.

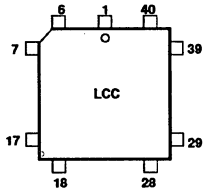
80CE654/83CE654



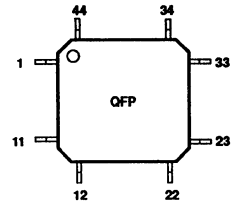
Pin	Function	Pin	Function
1	P1.5	23	P2.5/A13
2	P1.6/SCL	24	P2.6/A14
3	P1.7/SDA	25	P2.7/A15
4	RST	26	PSEN
5	P3.0/RxD	27	ALE
6	VSS4	28	VSS2
7	P3.1/TxD	29	EA
8	P3.2/INT0	30	P0.7/AD7
9	P3.3/INT1	31	P0.6/AD6
10	P3.4/T0	32	P0.5/AD5
11	P3.5/T1	33	P0.4/AD4
12	P3.6/WR	34	P0.3/AD3
13	P3.7/RD	35	P0.2/AD2
14	XTAL2	36	P0.1/AD1
15	XTAL1	37	P0.0/AD0
16	VSS1	38	VCC2
17	VCC1	39	VSS3
18	P2.0/A8	40	P1.0
19	P2.1/A9	41	P1.1
20	P2.2/A10	42	P1.2
21	P2.3/A11	43	P1.3
22	P2.4/A12	44	P1.4

8032AH/8052AH, 80C32/80C52/87C52, 83C524/87C524*, 80C528/83C528/87C528*





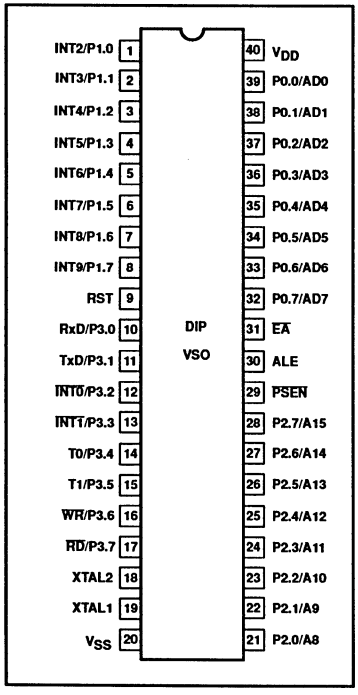
Pin	Function	Pin	Function
1	NC	23	NC
2	T2/P1.0	24	P2.0/A8
3	T2EX/P1.1	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6	30	P2.6/A14
9	P1.7	31	P2.7/A15
10	RST	32	PSEN
11	RxD/P3.0	33	ALE/PROG
12	NC	34	NC
13	TxD/P3.1	35	EA/VPP
14	INT0/P3.2	36	P0.7/AD7
15	INT1/P3.3	37	P0.6/AD6
16	T0/P3.4	38	P0.5/AD5
17	T1/P3.5	39	P0.4/AD4
18	WR/P3.6	40	P0.3/AD3
19	RD/P3.7	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	VSS	44	VCC



Pin	Function	Pin	Function
1	P1.5	23	P2.5/A13
2	P1.6	24	P2.6/A14
3	P1.7	25	P2.7/A15
4	RST	26	PSEN
5	RxD/P3.0	27	ALE/PROG
6	NC	28	NC
7	TxD/P3.1	29	EA/VPP
8	INT0/P3.2	30	P0.7/AD7
9	INT1/P3.3	31	P0.6/AD6
10	T0/P3.4	32	P0.5/AD5
11	T1/P3.5	33	P0.4/AD4
12	WR/P3.6	34	P0.3/AD3
13	RD/P3.7	35	P0.2/AD2
14	XTAL2	36	P0.1/AD1
15	XTAL1	37	P0.0/AD0
16	VSS	38	VCC
17	NC	39	NC
18	P2.0/A8	40	T2/P1.0
19	P2.1/A9	41	T2EXP/P1.1
20	P2.2/A10	42	P1.2
21	P2.3/A11	43	P1.3
22	P2.4/A12	44	P1.4

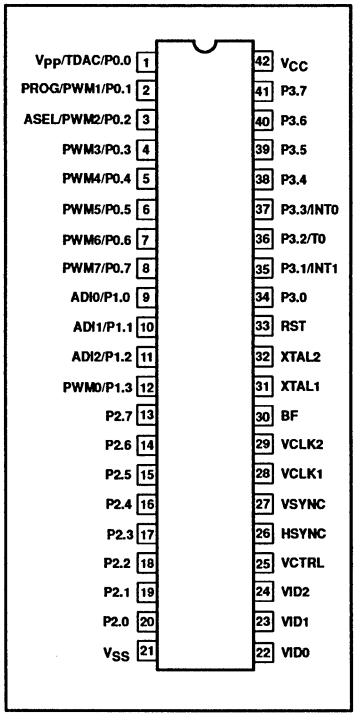
* P1.6 and P1.7 have the alternate functions SCL and SDA, respectively, on the 83C524/87C524 and 80C528/83C528/87C528.

80CL31/80CL51, 80CL410/83CL410*

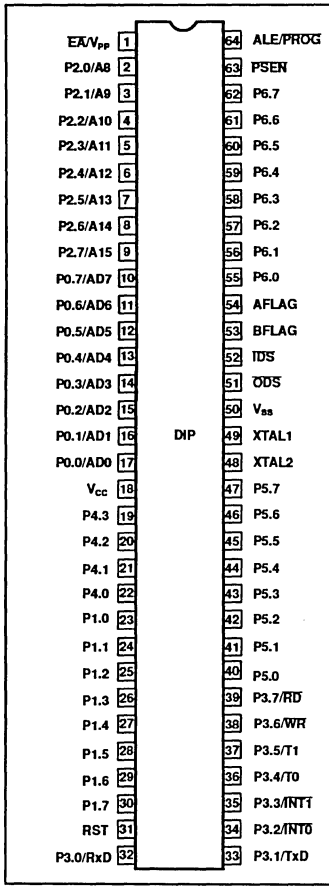


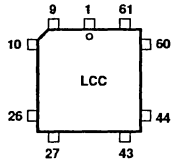
* P1.6 and P1.7 have the alternate functions SCL and SDA, respectively, on the 80CL410/83CL410.

83C053/83C054/87C054



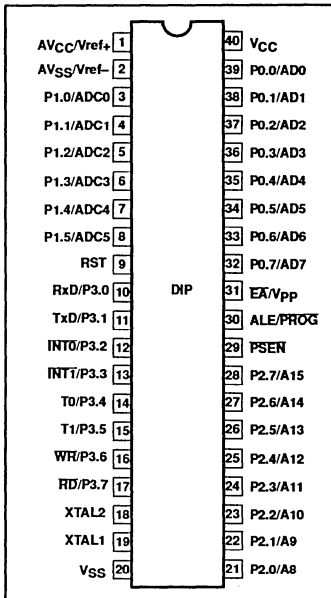
80C451/83C451/87C451

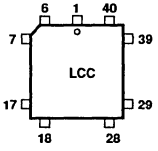




Pin	Function	Pin	Function
1	EA/V_{PP}	35	RST
2	P2.0/A8	36	P3.0/RxD
3	P2.1/A9	37	P3.1/TxD
4	P2.2/A10	38	P3.2/INT0
5	P2.3/A11	39	P3.3/INT1
6	P2.4/A12	40	P3.4/T0
7	P2.5/A13	41	P3.5/T1
8	P2.6/A14	42	P3.6/WR
9	P2.7/A15	43	P3.7/RD
10	P0.7/AD7	44	P5.0
11	P0.6/AD6	45	P5.1
12	P0.5/AD5	46	P5.2
13	P0.4/AD4	47	P5.3
14	P0.3/AD3	48	P5.4
15	P0.2/AD2	49	P5.5
16	P0.1/AD1	50	P5.6
17	P0.0/AD0	51	P5.7
18	V_{CC}	52	XTAL2
19	P4.7	53	XTAL1
20	P4.6	54	V_{SS}
21	P4.5	55	ODS
22	P4.4	56	IDS
23	P4.3	57	BFLAG
24	P4.2	58	AFLAG
25	P4.1	59	P6.0
26	P4.0	60	P6.1
27	P1.0	61	P6.2
28	P1.1	62	P6.3
29	P1.2	63	P6.4
30	P1.3	64	P6.5
31	P1.4	65	P6.6
32	P1.5	66	P6.7
33	P1.6	67	PAEN
34	P1.7	68	ALE/PROG

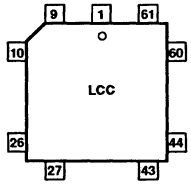
80C550/83C550/87C550



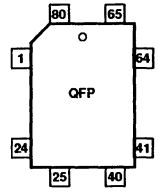


Pin	Function	Pin	Function	Pin	Function
1	AV_{CC}	16	P3.2/INT0	31	P2.6/A14
2	Vref+	17	P3.3/INT1	32	P2.7/A15
3	Vref−	18	P3.4/T0	33	PSEN
4	AV_{SS}	19	P3.5/T1	34	ALE/PROG
5	P1.0/ADC0	20	P3.6/WR	35	EA/V_{PP}
6	P1.1/ADC1	21	P3.7/RD	36	P0.7/AD7
7	P1.2/ADC2	22	XTAL2	37	P0.6/AD6
8	P1.3/ADC3	23	XTAL1	38	P0.5/AD5
9	P1.4/ADC4	24	V_{SS}	39	P0.4/AD4
10	P1.5/ADC5	25	P2.0/A8	40	P0.3/AD3
11	P1.6/ADC6	26	P2.1/A9	41	P0.2/AD2
12	P1.7/ADC7	27	P2.2/A10	42	P0.1/AD1
13	RST	28	P2.3/A11	43	P0.0/AD0
14	P3.0/RxD	29	P2.4/A12	44	V_{CC}
15	P3.1/TxD	30	P2.5/A13		

80C552/83C552/87C552*, 80C562/83C562



Pin	Function	Pin	Function
1	P5.0/ADC0	35	XTAL1
2	V_{DD}	36	V_{SS}
3	STADC	37	V_{SS}
4	$\overline{PWM0}$	38	NC
5	$\overline{PWMT}$	39	P2.0/A08
6	$\overline{EW}$	40	P2.1/A09
7	P4.0/CMSR0	41	P2.2/A10
8	P4.1/CMSR1	42	P2.3/A11
9	P4.2/CMSR2	43	P2.4/A12
10	P4.3/CMSR3	44	P2.5/A13
11	P4.4/CMSR4	45	P2.6/A14
12	P4.5/CMSR5	46	P2.7/A15
13	P4.6/CMT0	47	$\overline{PSEN}$
14	P4.7/CMT1	48	ALE/$\overline{PROG}$
15	RST	49	$\overline{EA}$/V_{PP}
16	P1.0/CT0I	50	P0.7/AD7
17	P1.1/CT1I	51	P0.6/AD6
18	P1.2/CT2I	52	P0.5/AD5
19	P1.3/CT3I	53	P0.4/AD4
20	P1.4/T2	54	P0.3/AD3
21	P1.5/RT2	55	P0.2/AD2
22	P1.6/SCL	56	P0.1/AD1
23	P1.7/SDA	57	P0.0/AD0
24	P3.0/RxD	58	AVref–
25	P3.1/TxD	59	AVref+
26	P3.2/$\overline{INT0}$	60	AV_{SS}
27	P3.3/$\overline{INT1}$	61	AV_{DD}
28	P3.4/T0	62	P5.7/ADC7
29	P3.5/T1	63	P5.6/ADC6
30	P3.6/$\overline{WR}$	64	P5.5/ADC5
31	P3.7/$\overline{RD}$	65	P5.4/ADC4
32	NC	66	P5.3/ADC3
33	NC	67	P5.2/ADC2
34	XTAL2	68	P5.1/ADC1

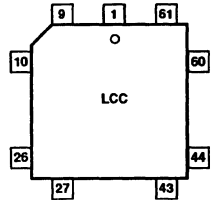


Pin	Function	Pin	Function
1	P4.1/CMSR1	41	P2.3/A11
2	P4.2/CMSR2	42	P2.4/A12
3	NC	43	NC
4	P4.3/CMSR3	44	NC
5	P4.4/CMSR4	45	P2.5/A13
6	P4.5/CMSR5	46	P2.6/A14
7	P4.6/CMT0	47	P2.7/A15
8	P4.7/CMT1	48	$\overline{PSEN}$
9	RST	49	ALE/$\overline{PROG}$
10	P1.0/CT0I	50	$\overline{EA}$/V_{PP}
11	P1.1/CT1I	51	P0.7/AD7
12	P1.2/CT2I	52	P0.6/AD6
13	P1.3/CT3I	53	P0.5/AD5
14	P1.4/T2	54	P0.4/AD4
15	P1.5/RT2	55	P0.3/AD3
16	P1.6/SCL	56	P0.2/AD2
17	P1.7/SDA	57	P0.1/AD1
18	P3.0/RxD	58	P0.0/AD0
19	P3.1/TxD	59	AVref–
20	P3.2/$\overline{INT0}$	60	AVref+
21	NC	61	AV_{SS}
22	NC	62	NC
23	P3.3/$\overline{INT1}$	63	AV_{DD}
24	P3.4/T0	64	P5.7/ADC7
25	P3.5/T1	65	P5.6/ADC6
26	P3.6/$\overline{WR}$	66	P5.5/ADC5
27	P3.7/$\overline{RD}$	67	P5.4/ADC4
28	NC	68	P5.3/ADC3
29	NC	69	P5.2/ADC2
30	NC	70	P5.1/ADC1
31	XTAL2	71	P5.0/ADC0
32	XTAL1	72	V_{DD}
33	IC	73	IC
34	V_{SS}	74	STADC
35	V_{SS}	75	$\overline{PWM0}$
36	V_{SS}	76	$\overline{PWMT}$
37	NC	77	$\overline{EW}$
38	P2.0/A08	78	NC
39	P2.1/A09	79	NC
40	P2.2/A10	80	P4.0/CMSR0

NC = not connected
IC = internally connected (do not use)

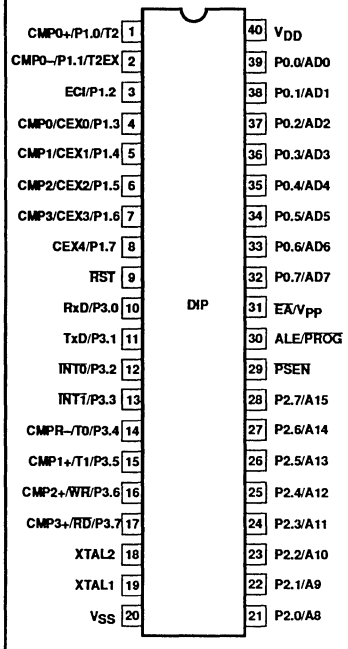
* P1.6 and P1.7 have the alternate functions SCL and SDA, respectively, on the 80C552/83C552/87C552.

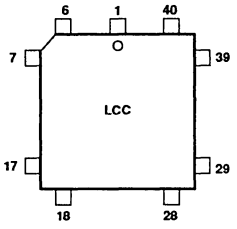
80C592/83C592/87C592



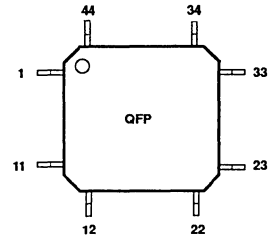
Pin	Function	Pin	Function
1	P5.0/ADC0	35	V_{SS}
2	V_{DD}	36	P2.0/A08
3	STADC	37	P2.1/A09
4	$\overline{PWM0}$	38	P2.2/A10
5	$\overline{PWMT}$	39	P2.3/A11
6	$\overline{EW}$	40	P2.4/A12
7	P4.0/CMSR0	41	P2.5/A13
8	P4.1/CMRS1	42	P2.6/A14
9	P4.2CMSR2	43	P2.7/A15
10	P4.3/CMSR3	44	$\overline{PSEN}$
11	P4.4/CMSR4	45	ALE/$\overline{PROG}$
12	P4.5/CMSR5	46	$\overline{EA}$/V_{PP}
13	P4.6/CMT0	47	P0.7/AD7
14	P4.7/CMT1	48	P0.6/AD6
15	RST	49	P0.5/AD5
16	P1.0/CT0I/INT2	50	P0.4/AD4
17	P1.1/CT1I/INT3	51	P0.3/AD3
18	P1.2/CT2I/INT4	52	P0.2/AD2
19	P1.3/CT3I/INT5	53	P0.1/AD1
20	P1.4/T2	54	P0.0/AD0
21	P1.5/RT2	55	REF
22	CV_{SS}	56	CRX1
23	P1.6/CTX0	57	CRX0
24	P1.7/CTX1	58	AVref–
25	P3.0/RxD	59	AVref+
26	P3.1TxD	60	AV_{SS}
27	P3.2/$\overline{INT0}$	61	AV_{DD}
28	P3.3/$\overline{INT1}$	62	P5.7/ADC7
29	P3.4/T0	63	P5.6/ADC6
30	P3.5/T1	64	P5.5/ADC5
31	P3.6/$\overline{WR}$	65	P5.4/ADC4
32	P3.7/$\overline{RD}$	66	P5.3/ADC3
33	XTAL2	67	P5.2/ADC2
34	XTAL1	68	P5.1/ADC1

80C575/83C575/87C575



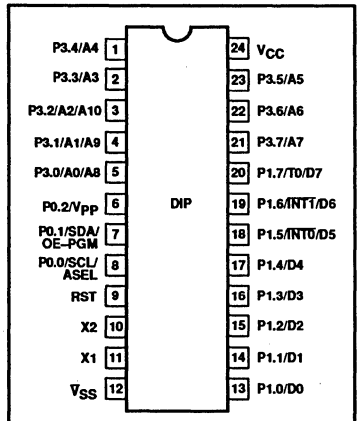


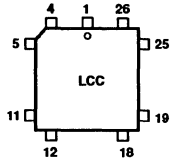
Pin	Function	Pin	Function
1	NC	23	NC
2	T2/P1.0/CMP0+	24	P2.0/A8
3	T2EX/P1.1/CMP0–	25	P2.1/A9
4	P1.2/ECI	26	P2.2/A10
5	P1.3/CMP0/CEX0	27	P2.3/A11
6	P1.4/CMP1/CEX1	28	P2.4/A12
7	P1.5/CMP2/CEX2	29	P2.5/A13
8	P1.6/CMP3/CEX3	30	P2.6/A14
9	P1.7/CEX4	31	P2.7/A15
10	RST	32	PSEN
11	RxD/P3.0	33	ALE/PROG
12	NC	34	NC
13	TxD/P3.1	35	EA/VPP
14	INT0/P3.2	36	P0.7/AD7
15	INT1/P3.3	37	P0.6/AD6
16	T0/P3.4/CMPR–	38	P0.5/AD5
17	T1/P3.5/CMP1+	39	P0.4/AD4
18	WR/P3.6/CMP2+	40	P0.3/AD3
19	RD/P3.7/CMP3+	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	VSS	44	VCC



Pin	Function	Pin	Function
1	P1.5/CMP2/CEX2	23	P2.5/A13
2	P1.6/CMP3/CEX3	24	P2.6/A14
3	P1.7/CEX4	25	P2.7/A15
4	RST	26	PSEN
5	RxD/P3.0	27	ALE/PROG
6	NC	28	NC
7	TxD/P3.1	29	EA/VPP
8	INT0/P3.2	30	P0.7/AD7
9	INT1/P3.3	31	P0.6/AD6
10	T0/P3.4/CMPR–	32	P0.5/AD5
11	T1/P3.5/CMP1+	33	P0.4/AD4
12	WR/P3.6/CMP2+	34	P0.3/AD3
13	RD/P3.7/CMP3+	35	P0.2/AD2
14	XTAL2	36	P0.1/AD1
15	XTAL1	37	P0.0/AD0
16	VSS	38	VCC
17	NC	39	NC
18	P2.0/A8	40	T2/P1.0/CMP0+
19	P2.1/A9	41	T2EX/P1.1/CMP0–
20	P2.2/A10	42	P1.2/ECI
21	P2.3/A11	43	P1.3/CMP0/CEX0
22	P2.4/A12	44	P1.4/CMP1

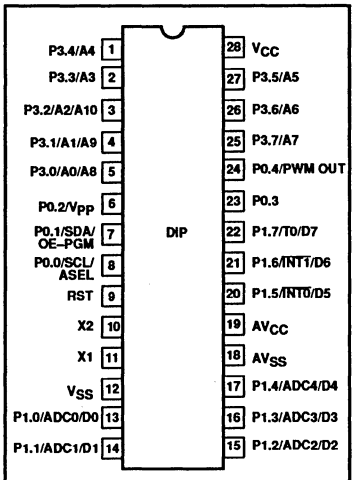
83C751/87C751

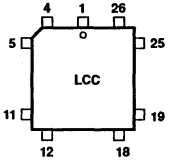




Pin	Function	Pin	Function
1	P3.4/A4	15	P1.0/D0
2	P3.3/A3	16	P1.1/D1
3	P3.2/A2/A10	17	P1.2/D2
4	P3.1/A1/A9	18	P1.3/D3
5	N.C.	19	P1.4/D4
6	P3.0/A0/A8	20	P1.5/INT0/D5
7	P0.2/V_{PP}	21	N.C.
8	P0.1/SDA/OE–PGM	22	N.C.
9	P0.0/SCL/ASEL	23	P1.6/INT1/D6
10	N.C.	24	P1.7/T0/D7
11	RST	25	P3.7/A7
12	X2	26	P3.6/A6
13	X1	27	P3.5/A5
14	V_{SS}	28	V_{CC}

83C752/87C752





Pin	Function	Pin	Function
1	P3.4/A4	15	P1.2/ADC2/D2
2	P3.3/A3	16	P1.3/ADC3/D3
3	P3.2/A2/A10	17	P1.4/ADC4/D4
4	P3.1/A1/A9	18	AV_{SS}
5	P3.0/A0/A8	19	AV_{CC}
6	P0.2/V_{PP}	20	P1.5/INT0/D5
7	P0.1/SDA/OE–PGM	21	P1.6/INT1/D6
8	P0.0/SCL/ASEL	22	P1.7/T0/D7
9	RST	23	P0.3
10	X2	24	P0.4/PWM OUT
11	X1	25	P3.7/A7
12	V_{SS}	26	P3.6/A6
13	P1.0/ADC0/D0	27	P3.5/A5
14	P1.1/ADC1/D1	28	V_{CC}

Philips — a worldwide company

Argentina: PHILIPS ARGENTINA S.A. Div. Philips Components, Vedia 3892, 1430 BUENOS AIRES, Tel. (01) 541-4261

Australia: PHILIPS COMPONENTS PTY Ltd, 34 Waterloo Road, NORTH RYDE, NSW 2113, TEL. (02) 805-4455. Fax (02) 805 4466

Austria: OSTERREICHISCHE PHILIPS INDUSTRIE G.m.b.H., UB Bauelemente, Triester Str. 64, 1101 WIEN. Tel. (0222) 60 101-820

Belgium: N.V. PHILIPS PROF. SYSTEMS — Components Div., 80 Rue Des Deux Gares, B-1070 BRUXELLES, Tel. (02) 5256111

Brazil: PHILIPS COMPONENTS (Active Devices & LCD) Rua Do Rocio 220, SAO PAULO-SP, CEP 4552, P.O. Box 7383, CEP 01051, Tel. (011) 829-1166. Fax (011) 829-1849. PHILIPS COMPONENTS (Passive Devices & Materials) Av. Francisco Monteiro 702, RIBEIRAO PIRES-SP, CEP 09400, Tel. (011) 459-8211. Fax (011) 459-8282

Canada: PHILIPS ELECTRONICS LTD., Philips Components, 601 Milner Ave., SCARBOROUGH, Ontario, M1B 1M8 Tel. (416) 292-5161. (IC Products) SIGNETICS CANADA LTD., 1 Eva Road, Suite 411, ETOBICOKE, Ontario, M9C 4Z5, Tel. (416) 626-6676

Chile: PHILIPS CHILENA S.A., Av. Santa Maria, Casilla 0760, SANTIAGO, Tel. (02) 077 38 16

Colombia: IPRELENSO LTDA., Carrera 21 No. 56-17, BOGOTA, D.E., P.O. Box 77621, Tel. (01) 2497624

Denmark: PHILIPS COMPONENTS A/S, Prags Boulevard 80, PB1919, DK-2300 COPENHAGEN S, Tel. 32-883333

Finland: PHILIPS COMPONENTS, Sinikalliontie 3, SF-02630 ESPOO Tel. 358-0-50261

France: PHILIPS COMPOSANTS, 117 Quai du President Roosevelt, 92134 ISSY-LES-MOULINEAUX Cedex, Tel. (01) 40938000. Fax. 01 409 38692

Germany: PHILIPS COMPONENTS UB der Philips G.m.b.H., Burchardstrasse 19, D-2 HAMBURG 1, Tel. (040) 3296-0, Fax. 040 32969 12.

Greece: PHILIPS HELLENIQUE S.A., Components Division, No. 15, 25th March Street, GR 17778 TAVROS, Tel. (01) 4894339/4894911

Hong Kong: PHILIPS HONG KONG LTD., Components Div., 15/F Philips Ind. Bldg., 24 Kung Yip St., KWAI CHUNG, Tel. (0)-4245 121

India: PEICO ELECTRONICS & ELECTRICALS LTD., Components Dept., Shivsagar Estate 'A' Block, P.O. Box 6598, 254-D Dr. Annie Besant Rd., BOMBAY-40018 Tel. (022) 4921500–4921515. Fax. 022 494 19063

Indonesia: P.T. PHILIPS-RALIN ELECTRONICS, Components Div., Setiabudi II Building, 6th Fl., Jalan H.R. Rasuna Said (P.O. Box 223/KBY) Kuningan, JAKARTA, 12910, Tel. (021) 517995

Ireland: PHILIPS ELECTRONICS (IRELAND) LTD., Components Division, Newstead, Clonskeagh, DUBLIN 14, Tel. (01) 693355

Italy: PHILIPS S.p.A., Philips Components, Piazza IV Novembre 3, I-20124 MILANO, Tel. (02) 6752.1, Fax. 02 67522642

Japan: PHILIPS JAPAN LTD., Components Division, Philips Bldg. 13-37, Kohnan 2-chome, Minato-ku, TOKYO 108, Tel. (03) 813-3740-5030. Fax. 03 81337400570

Korea (Republic of): PHILIPS INDUSTRIES (KOREA) LTD., Components Division, Philips House, 260-199 Itaewon-dong, Yongsan-ku, SEOUL, Tel. (02) 794-5011

Malaysia: PHILIPS MALAYSIA SDN BHD, Components Div., 3 Jalan SS15/2A SUBANG, 47500 PETALING JAYA, Tel. (03) 7345511

Mexico: PHILIPS COMPONENTS, Paseo Triunfo de la Republica, No 215 Local 5, Cd Juarez CHIHUAHUA 32340 MEXICO, Tel. (16) 18-67-01/02

Netherlands: PHILIPS NEDERLAND B.V., Marktgroep Philips Components, Postbus 90050, 5600 PB EINDHOVEN, Tel. (040) 783749

New Zealand: PHILIPS NEW ZEALAND LTD., Components Division, 2 Wagener Place, C.P.O. Box 1041, AUCKLAND, Tel. (09) 894-160, Fax. (09) 897-811

Norway: NORSK A/S PHILIPS, Philips Components, Box 1, Manglerud 0612, OSLO, Tel. (02) 74 10 10

Pakistan: PHILIPS ELECTRICAL CO. OF PAKISTAN LTD., Philips Markaz, M.A. Jinnah Rd., KARACHI-3, Tel. (021) 725772

Peru: CADESA, Carretera Central 6.500, LIMA 3, Apartado 5612, Tel. 51-14-350059

Philippines: PHILIPS SEMICONDUCTORS PHILIPPINES Inc., 106 Valero St. Salcedo Village, P.O. Box 911, MAKATI, Metro MANILA, Tel. (63-2) 810-0161 Fax. 632 817 3474

Portugal: PHILIPS PORTUGUESA S.A.R.L., Av. Eng. Duarte Pacheco 6, 1009 LISBOA Codex, Tel. (019) 68 31 21

Singapore: PHILIPS SINGAPORE, PTE LTD., Components Div., Lorong 1, Toa Payoh, SINGAPORE 1231, Tel. 35 02 000

South Africa: S.A. PHILIPS PTY LTD., Components Division, 195-215 Main Road, JOHANNESBURG 2000, P.O. Box 7430, Tel. (011) 470-5434. Fax. (011) 470 5494

Spain: PHILIPS COMPONENTS, Balmes 22, 08007 BARCELONA, Tel. (03) 301 6312. Fax. 03 301 4243

Sweden: PHILIPS COMPONENTS, A.B., Tegeluddsvägen 1, S-11584 STOCKHOLM, Tel. (0)8-7821000

Switzerland: PHILIPS A.G., Components Dept., Allmendstrasse 140-142, CH-8027 ZÜRICH, Tel. (01) 4882211

Taiwan: PHILIPS TAIWAN LTD., 581 Min Sheng East Road, P.O. Box 22978, TAIPEI 10446, Taiwan, Tel. 886-2-509-7666. Fax. 886 2 500 5899

Thailand: PHILIPS ELECTRICAL CO. OF THAILAND LTD., 283 Silon Road, P.O. Box 961, BANGKOK, Tel. (02) 233-6330-9

Turkey: TÜRK PHILIPS TICARET A.S., Philips Components, Talatpasa Cad. No. 5, 80640 LEVENT/ISTANBUL, Tel. (01) 179 2770

United Kingdom: PHILIPS COMPONENTS LTD., Mullard House, Torrington Place, LONDON WC1E 7HD, Tel. (071) 580 6633, Fax. (071) 436 2196

United States: (Color Picture Tubes - Monochrome & Colour Display Tubes) PHILIPS DISPLAY COMPONENTS COMPANY, 1600 Huron Parkway, P.O. Box 963, ANN ARBOR, Michigan 48106, Tel. (313) 996-9400 (IC Products) SIGNETICS COMPANY, 811 East Arques Avenue, SUNNYVALE, CA 94088-3409, Tel. (800) 227-1817, Ext. 900 (Passive Components, Discrete Semiconductors, Materials and Professional Components & LCD) PHILIPS SEMICONDUCTORS, Discrete Products Division, 2001 West Blue Heron Blvd., P.O. Box 10330, RIVIERA BEACH, Florida 33404, Tel. (407) 881-3200

Uruguay: PHILIPS COMPONENTS, Coronel Mora 433, MONTEVIDEO, Tel. (02) 70-4044

Venezuela: MAGNETICA S.A., Calle 6, Ed. Las Tres Jotas, CARACAS 1074A, App. Post. 78117, Tel. (02) 241 7509

Zimbabwe: PHILIPS ELECTRICAL (PVT) LTD., 62 Mutare Road, HARARE, P.O. Box 994, Tel. 47211

For all other countries apply to: Philips Components, Marketing Communications, P.O. Box 218, 5600 MD EINDHOVEN, The Netherlands, Telex 35000 phtcnl, Fax +31-40-724825

ADS92 ©Philips Export B.V. 1992

All rights are reserved. Reproduction in whole or in part is prohibited without the prior written consent of the copyright owner.

The information presented in this document does not form part of any quotation or contract, is believed to be accurate and reliable and may be changed without notice. No liability will be accepted by the publisher for any consequence of its use. Publication thereof does not convey nor imply any license under patent — or industrial or intellectual property rights.

11/25/91

98-8080-390-01

2083M/50M/CR1/0292 1040pp

Philips Semiconductors

PHILIPS

PHILIPS