price $10.00

MC14 1000 series

microcomputer programming reference manual

MOTOROLA

MC141000 Series PROGRAMMING REFERENCE MANUAL

TABLE OF CONTENTS

ILLUSTRATIONS

LIST OF ABBREVIATIONS

A	Accumulator
ALU	Arithmetic Logic Unit
CKI	Constant and K-Input Logic
CL	Call Latch
I(B)	Bit Field of Instruction
I(C)	Constant Field of Instruction
I(W)	Branch Address of Instruction
Ki	K inputs
LSB	Least Significant Bit
LSD	Least Significant Digit
MSB	Most Significant Bit
MSD	Most Significant Digit
M(X,Y)	RAM Memory Location = X Address (0 to 7), Y Address (0 to 15)
M(X,Y,B)	RAM Memory Bit Location (B = 0, 1, 2, or 3)
O	Output Register
PA	ROM Page Address Register
PB	ROM Page Buffer Register
PC	Program Counter
PLA	Programmable Logic Array
PLAIR	PLA Input Register
R	R-Output Register
R(Y)	R-Output Latch Y
RAM	Random Access Memory (Read/Write)
ROM	Read Only Memory
S	Status
SL	Status Latch
SRR	Subroutine Return Register
X	RAM X Address Register
Y	RAM Y Address Register

INTRODUCTION

MC141000/1200/1099 Microcomputers

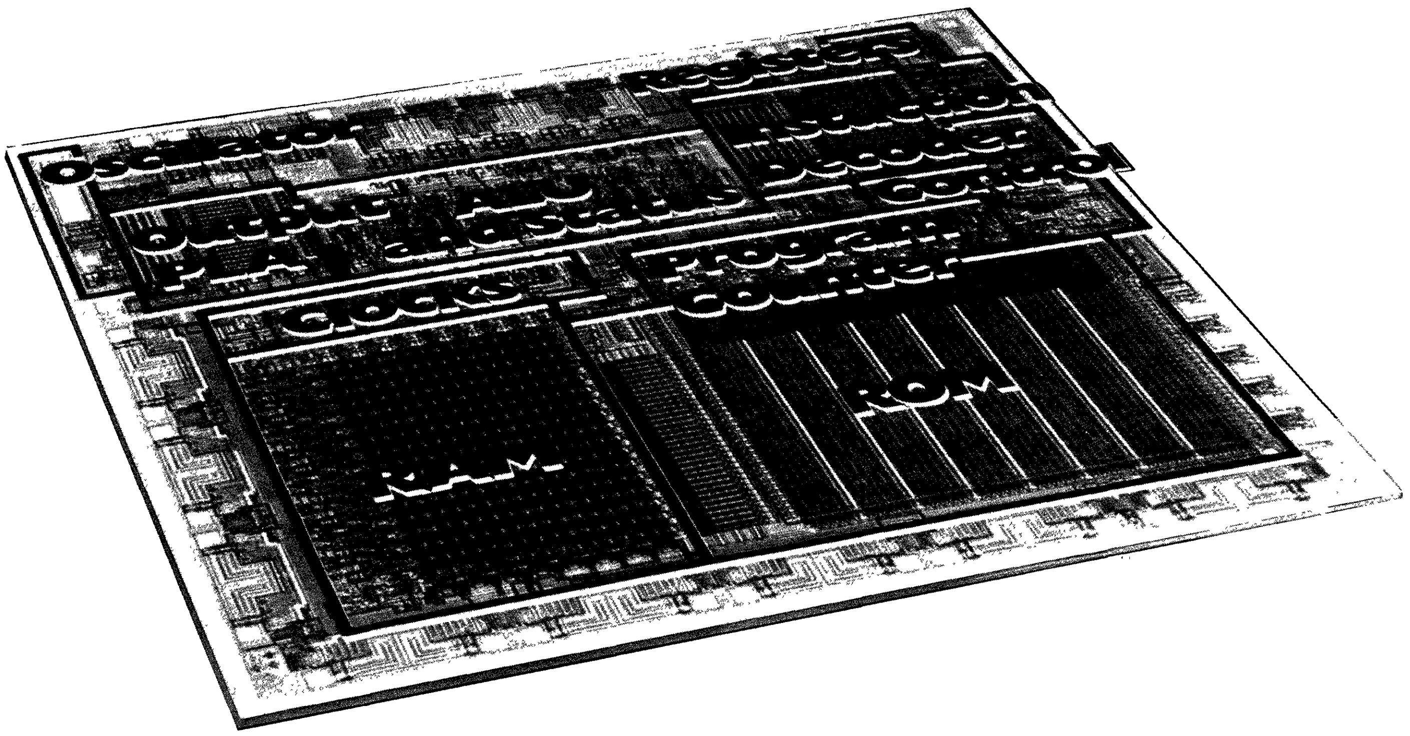


Figure 1. The MC141000 – A Single-Chip Microcomputer featuring CMOS Technology.

The MC141000 and MC141200 single-chip microcomputers are complete four-bit microcomputers which contain read-only program memory, random-access memory, arithmetic logical unit, buffered inputs, and output drivers.

The MC141200 has 16 outputs (R lines) in a 40-pin package.

The MC141000 is a limited pinout version with 11 R outputs in a 28-pin package.

Electrical and mechanical specifications can be found on the MC141000 and MC141200 Data Sheet.

The MC141099 is a 48-pin version of the MC141200, intended primarily as a design tool for prototyping and debugging MC141000/1200 systems prior to manufacture. As such, it is designed for use with external programmable logic (PLA) and memory and does not have these functions on-chip. The larger number of package pins permits interfacing with these external functions.

Microprogramming

If the standard instruction set, Page 11, is found to be inadequate to meet the requirements of a specific application, the instruction set may be modified to a limited extent by redefining the instruction-decode programmable logic array. Please consult the Motorola MOS Facility in Austin, Texas for details.

MC14100 Family Support Products

To support the MC141000 microcomputer family, Motorola offers a complete hardware /software package for the development of a user system through system simulation, Figure 2. The software package consists of a cross-assembler, a loader and debugging capability. The hardware consists of a dedicated component board (MEX141000M) that interfaces with Motorola's standard EXORciser*, permitting its use as an MC14100 system simulator. The system runs under the control of the MDOS operating system and provides powerful development and debug capability prior to commitment to final production masks.

The debug phase of program development is expedited by use of the SIMULATOR breakpoint, trace, and single-step features, and the relative ease of program changes on a disk-based development system. Detailed information concerning this system is available in the MC141000 CROSS-ASSEMBLER MANUAL and the MC141000 SIMULATOR MODULE MANAUL.

A hardware approach to system development and debug can be implemented with the MC141099 processor. It is identical to the MC141200 with the exceptions that the program storage memory and output PLA are external, Figure 3. RAM, PROM or EPROM memory can be used for program storage and a PLA decoder or random logic can be used to simulate the output PLA.

For those who prefer to purchase a turnkey device, Motorola offers an in-house design, programming and development capability. Figure 4 outlines a typical development program from software specification to device production. Your local sales representative can supply cost and scheduling information for this service.

*EXORciser is a trademark of Motorola Inc.

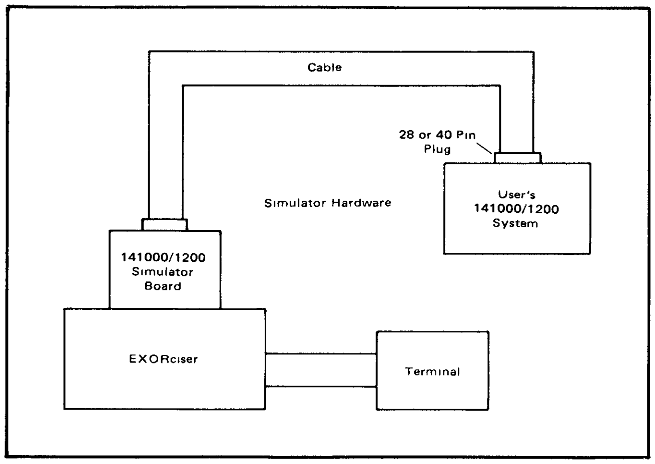


Figure 2 – EXORciser Based Software Development System

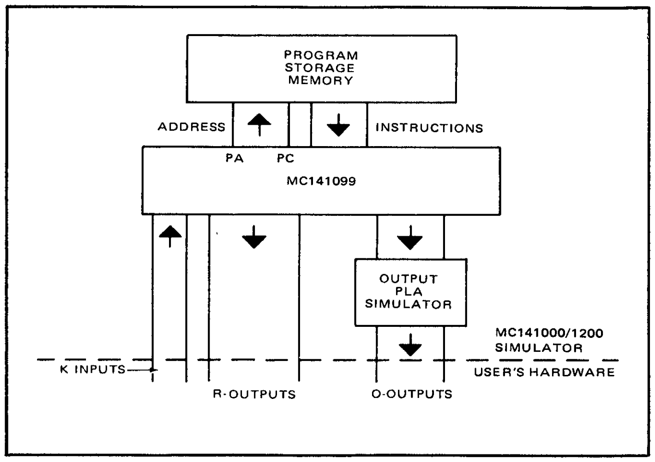


Figure 3 – MC141099 Based Software Development and Debug System

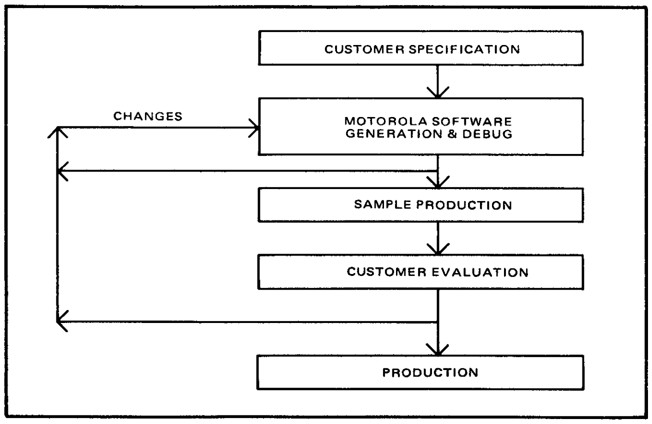


Figure 4 – Motorola Custom Software and Debug Development Schedule

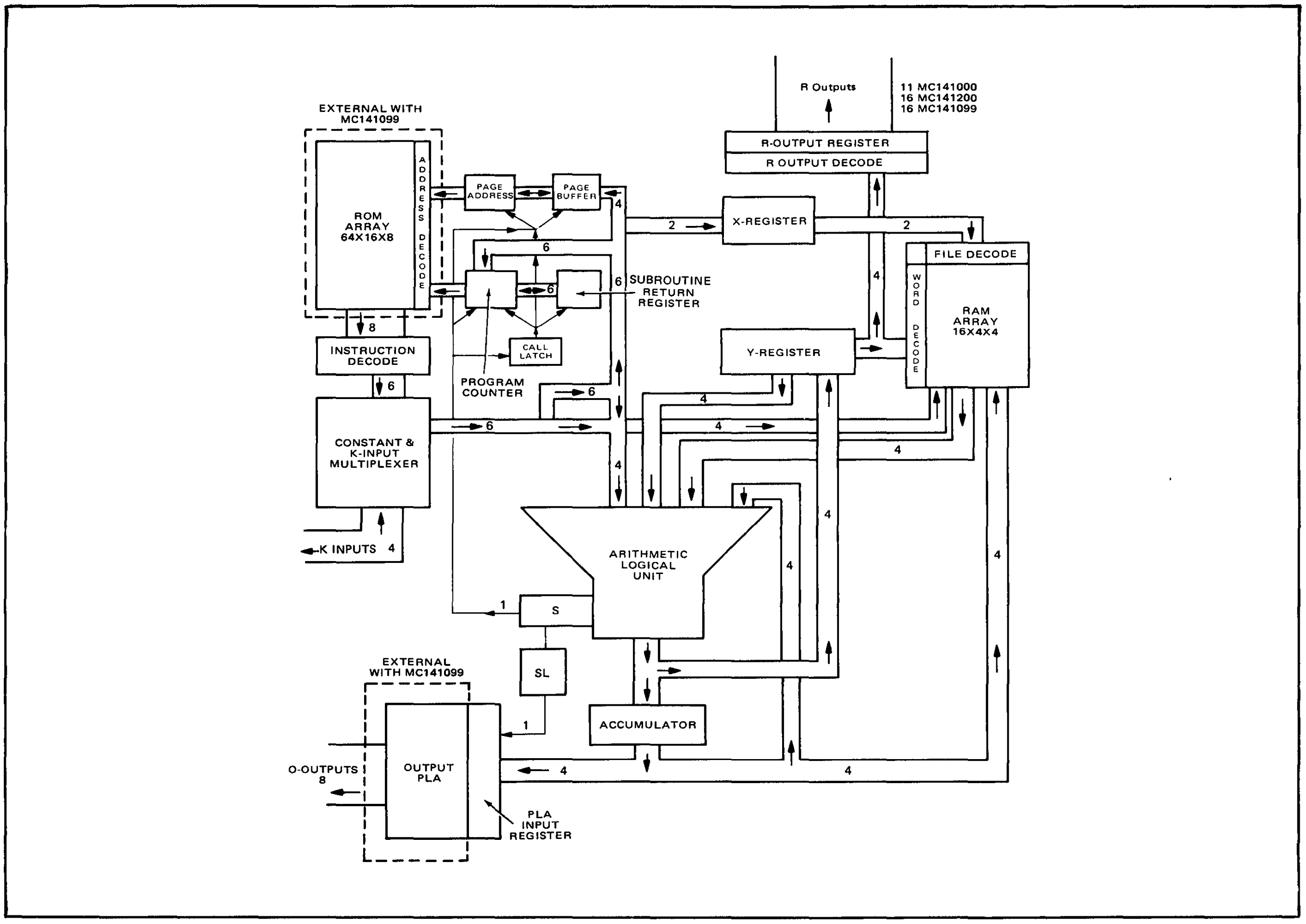


Figure 5 — Functional Block Diagram — MC141000/1200/1099

MC141000 MICROCOMPUTER RESOURCES

FUNCTIONAL BLOCKS

Figure 5 shows a block diagram of the resources available to the MC141000/1200/1099 programmer. They are:

A	The accumulator is used to store the result of an ALU operation for subsequent operations.
ALU	The arithmetic logical unit perform calculation and decision-making tasks.
K INPUTS	The K lines are the data input port. Since there are only four input lines, they are usually multiplexed under control of the R lines using external hardware. The inputs are diode protected and have a pull-down resister of approximately 50 K ohms; therefore, open inputs are read as a logic low.
O OUTPUTS	The eight outputs of the PLA are connected to output drivers which comprise the O-outputs. These output drivers may be manufactured as open emitter, active sink, or push-pull at the user's option.
PLA	The output programmable logic array is user-defined to specify the state of each of the eight O-outputs for each of the 32 possible PLAIR outputs.
PLAIR	The programmable logic array input register is a five-bit latch which latches the four accumulator bits and output of the status latch.
RAM	Variable data is stored in the 64-word, 4-bit per word Random Access Memory. Data is accessed by decoding a 2-bit file address (X register) and 4-bit word address (Y register).
ROM ARRAY	The user's instructions are mask programmed into the Read Only Memory (ROM). Instructions are addressed by a page address register (PA) and program counter (PC). A single subroutine return register (SRR) and page buffer register (PB) permit subroutine calls to any location within the ROM.
R OUTPUTS	The output of the Y register is decoded to select one of the R-output lines which can then be set or reset under program control. The R lines are used as control lines to scan keyboards and displays, perform handshakes, and interface external logic. The R-outputs may be manufactured as open emitter, active sink, or push-pull at the user's option.
S	All branches and subroutine calls are dependent on the state of status logic. It may be set or reset on logical or arithmetic operations and is set by the remainder of the instructions.
SL	The status latch latches the state of the status logic in order to preserve it for subsequent O-output operations. NOTE: S and SL are NOT identical.
Y REGISTER	The Y register is a multipurpose register used to address a word in a RAM file, to select an R output for manipulation by subsequent instructions, or as a general purpose counting and storage register.

Status and Status Latch

All program-modifying instructions (BRanch or CALL) are conditional on the state of the status logic. If status is set, the BRanch or CALL is executed by jumping to the ROM address specified by the operand field of the BRanch or CALL instruction and the contents of the page buffer register (PB). If status is reset, the BRanch or CALL is not to be taken, and the instruction following the BRanch or CALL is the next to execute. The BRanch or CALL takes six clock cycles (one instruction cycle) to execute whether the status is set or reset.

The status logic is normally set. Whenever it is reset, the reset condition only lasts for one instruction cycle and then returns to the set state. The only means to keep it reset for more than one instruction cycle is to execute more than one instruction in series which causes it to reset in series.

The status latch takes the state of the status logic and saves it during the execution of a YNEA instruction. Power on reset and INIT have no effect on the status latch. Therefore, a YNEA instruction must be executed before a TDO instruction to ensure that the desired state of status latch is placed into the PLA input register.

Power Up and Initialize

TABLE 1. Power Up and Initialize

	PC	PA	PB	CL	PLAIR	R OUTPUTS
Power Up	0	15	15	0	0	0
Initialize	0	$\overline{K}$	$\overline{K}$	0	0	0

When power is applied, the registers shown in Table 1 are loaded as shown for power up. All other internal registers and RAM come up in an arbitrary state.

After power is applied, the initialize (INIT) input may be used to reinitialize the processor. Internally, INIT has a 50 K ohm pull-down resistor which holds the INIT line low. It must be held high for a minimum of 6 full clock cycles and then returned to the low state. If a mechanical switch or other mechanical device is used to control INIT, it may be necessary to include a method of contact debounce to ensure a valid INIT pulse.

A valid INIT pulse will cause the registers to be loaded as shown in the table. The contents of registers other than those shown will remain unchanged during initialize. Note that the PA and PB are loaded with the 1's complement of the K-input lines (K_8 =MSB). This feature allows the MC141000 to be initialized to the first instruction on any page by controlling the K-inputs during initialization. This is useful where the same circuit may be used for several applications. Since the K-inputs have 50 K pull-down resistors, they will be a 0 (unless driven from another device) and the 1's complement (F) will be loaded into PA and PB.

MOTOROLA MC141000 ASSEMBLY FORMAT

Coding Format

Users who choose to develop their own software should provide assembler-compatible code conforming to the assembler input rules. The following syntax rules will allow user-generated source code to be assembled on any of the Motorola MC141000 cross-assemblers.

Input Format

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

|←— Label Field —→|←— Operator Field —→|←— Operand Field —→|← Comment Field →

Labels

Labels are a maximum of six alphanumeric characters. (Columns 7, 8 and 9 are not used.) The first character must be alphabetic. Labels may not contain imbedded blanks. Assembler mnemonics and directives must not be used as labels.

Valid characters are:
A through Z
0 through 9
=

Comments

An asterisk in column 1 indicates that the entire line is a comment.

Operands

Operands must be non-negative decimal constants or valid labels.

Assembler Directives

All versions of the Motorola MC141000 assembler support the following assembler directives (pseudo-ops). Assembler directives must start in COL 10.

ORG	nnnn	(Where nnnn is a decimal number) This causes the assembler to place the machine code at offset nnnn. Any number of ORGs may be used, but they must always be in ascending order.
EJECT		This causes the printer to go to top of form.
PAGE		Similar to ORG but causes the next instruction to be placed at location 00 of the next sequential ROM page.

label 1* BL	Label 2	"BRANCH LONG" Causes two machine instructions to be generated. First an LDP to the page of Label 2, than a BR to that label.
label 1* Call L	Label 2	"CALL LONG" Causes two machine instructions to be generated. First an LDP to the page of Label 2, then a BR to that label.

*Optional

Several versions of the cross-assembler are in use at Motorola and some have more extensive capabilities than those outlined above. For more information on source-code submittal, please consult the Motorola MOS Facility, Austin, Texas.

Additional Documentation

Detailed hardware specifications for the MC141000/1200/1099 are available in the data sheet, and information concerning the M6800 EXORciser-based MC141000/1200 development/debug system is contained in the MC141000 CROSS-ASSEMBLER MANUAL and the MC141000 SIMULATOR MODULE MANUAL. Your local distributor or Motorola sales representative can supply documentation.

INSTRUCTION TABLES

The MC141000 microcomputer instruction set consists of 43 standard instructions. These are summarized in Table 2, which lists the instructions by function, and Table 3, which lists the instructions alphabetically.

TABLE 2. MC141000/1200 Instruction Set, Functional Listing

Function	Mnemonic	Condition Setting Status	Action
ROM Addressing	BR	Always	See Fig. 7
	CALL	Always	See Fig. 7
	LDP	Always	$I(C) \rightarrow PB$
	RETN	Always	See Fig. 7
RAM X Addressing	COMX	Always	$\overline{X} \rightarrow X$
	LDX	Always	$I(B) \rightarrow X$
Output	CLO	Always	$O \rightarrow PLAIR$
	RSTR	Always	$O \rightarrow R(Y)$
	SETR	Always	$1 \rightarrow R(Y)$
	TDO	Always	$SL,A \rightarrow PLAIR$
Input	KNEZ	K-inputs not zero	$K \neq 0$
	TKA	Always	$K \rightarrow A$
Internal Data Transfer	CLA	Always	$0 \rightarrow A$
	TAM	Always	$A \rightarrow M(X,Y)$
	TAMIY	Always	$A \rightarrow M(X,Y), Y + 1 \rightarrow Y$
	TAMZA	Always	$A \rightarrow M(X,Y), 0 \rightarrow A$
	TAY	Always	$A \rightarrow Y$
	TCY	Always	$I(C) \rightarrow Y$
	TCMIY	Always	$I(C) \rightarrow M(X,Y), Y + 1 \rightarrow Y$
	TMA	Always	$M(X,Y) \rightarrow A$
	TMY	Always	$M(X,Y) \rightarrow Y$
	TYA	Always	$Y \rightarrow A$
	XMA	Always	$M(X,Y) \leftrightarrow A$
Bit Manipulation	RBIT	Always	$0 \rightarrow M(X,Y,B)$
	SBIT	Always	$1 \rightarrow M(X,Y,B)$
	TBIT1	Bit equal to 1	$M(X,Y,B) = 1$
Arithmetic	A6AAC	Carry	$A + 6 \rightarrow A$
	A8AAC	Carry	$A + 8 \rightarrow A$
	A10AA	Carry	$A + 10 \rightarrow A$
	AMAAC	Carry	$M(X,Y) + A \rightarrow A$
	CPAIZ	Carry	$\overline{A} + 1 \rightarrow A$
	DAN	Carry	$A - 1 \rightarrow A$
	DMAN	Carry	$M(X,Y) - 1 \rightarrow A$
	DYN	Carry	$Y - 1 \rightarrow Y$
	IA	Always	$A + 1 \rightarrow A$
	IMAC	Carry	$M(X,Y) + 1 \rightarrow A$
	IYC	Carry	$Y + 1 \rightarrow Y$
	SAMAN	If no borrow	$M(X,Y) - A \rightarrow A$
Logical	ALEC	Accumulator less than or equal to constant	$A \leq (C)$
	ALEM	Accumulator less than or equal to memory	$A \leq M(X,Y)$
	MNEZ	Memory not equal to zero	$M(X,Y) \neq 0$
	YNEA	Y-register not equal to accumulator	$Y \neq A,S \rightarrow SL$
	YNEC	Y-register not equal to constant	$Y \neq (C)$

TABLE 3. MC141000/1200 Standard Instruction Set, Alphabetical Listing

Opcode	Mnemonic	Description
0111 (C)	ALEC	If accumulator is less than or equal to I(C) field, status = 1.
00101001	ALEM	If accumulator is less than or equal to M(X,Y), status = 1.
00100101	AMAAC	Add memory to accumulator. Accumulator = result, status = carry.
00000110	A6AAC	Add 6 to accumulator. Accumulator = result, status = carry.
00000001	A8AAC	Add 8 to accumulator. Accumulator = result, status = carry.
00000101	A10AAC	Add 10 to accumulator. Accumulator = result, status = carry.
10 (W)	BR	Branch to label if status = 1.
11 (W)	CALL	Call subroutine if status = 1.
00101111	CLA	Clear contents of accumulator.
00001011	CLO	Clear PLA Input Register.
00000000	COMX	Complement X-Register.
00101101	CPAIZ	Complement accumulator, then add 1. If accumulator = 0, status = 1.
00000111	DAN	Decrement accumulator. If no borrow, status = 1.
00101010	DMAN	Load M(X, Y) into accumulator and decrement. If no borrow, status = 1.
00101100	DYN	Decrement Y-register. If no borrow, status = 1.
00001110	IA	Increment accumulator.
00101000	IMAC	Load M(X, Y) into accumulator and increment. Status = carry.
00101011	IYC	Increment Y-Register. Status = carry.
00001001	KNEZ	If K-inputs not equal to zero, status = 1.
0001 (C)	LDP	Load page buffer with I(C) field.
001111 (B)	LDX	Load X-register with I(B) field.
00100110	MNEZ	If M(X, Y) not equal to zero, status = 1.
001101 (B)	RBIT	Reset bit I(B) of M(X,Y).
00001111	RETN	Return from subroutine.
00001100	RSTR	Reset R-line specified by Y-register.
00100111	SAMAN	Subtract accumulator from memory. Accumulator = result. If no borrow, status = 1.
001100 (B)	SBIT	Set Bit I(B) of M(X,Y).
00001101	SETR	Set R-line specified by Y-register.
00000011	TAM	Transfer accumulator contents to M(X,Y).
00100000	TAMIY	Transfer accumulator contents to M(X,Y), increment Y-register.
00000100	TAMZA	Transfer accumulator contents to M(X,Y), zero accumulator.
00100100	TAY	Transfer accumulator contents to Y-register.
001110 (B)	TBIT1	If bit I(B) of M(X,Y) is one, status = 1.
0100 (C)	TCY	Transfer I(C) field to Y-register.
0110 (C)	TCMIY	Transfer I(C) field to M(X,Y), increment Y-register.
00001010	TDO	Transfer status latch and accumulator to O-output register.
00001000	TKA	Tranfer K-inputs to accumulator.
00100001	TMA	Transfer M(X,Y) to accumulator.
00100010	TMY	Transfer M(X,Y) to Y-register.
00100011	TYA	Transfer Y-register contents to accumulator.
00101110	XMA	Exchange contents of M(X,Y) and accumulator.
00000010	YNEA	If Y-register is not equal to accumulator, status and status latch = 1.
0101 (C)	YNEC	If Y-register is not equal to I(C) field, status = 1.

MACHINE OPERATION

The MC141000 microcomputer consists of 6 subsystems:

1) Read Only Memory (ROM)
2) Random Access Memory (RAM)
3) Output ports
4) Input port
5) Arithmetic Logical Unit (ALU)
6) The Instruction Decoder

The following sections will describe how each of these subsystems is controlled by the instruction set. Every instruction occupies a single memory byte and is executed in one instruction cycle (6 clock cycles).

ROM Array

The ROM consists of 8192 bits of mask-programmed memory organized as 1024 8-bit instructions. It is divided into 16 pages of 64 instructions per page.

Instructions within ROM are addressed by the page address register (PA) which contains the page number, and the program counter (PC) which contains the location of the instruction relative to the beginning of the page. The PC is incremented prior to fetching the next instruction (unless diverted by a BRanch or CALL) so each instruction is accessed in the numerical order of its address. A carry from the PC is not added to the PA so the program will "wraparound" within the page rather than executing the first instruction of the following page. Upon power up, the PC is set to zero and the PA and PB are set to 15.

Branch, Call and Return Operations — Figure 6

The normal sequence of instruction execution may be diverted by branch (BR) or subroutine call (CALL) instructions which are conditional on the state of the status logic. If the status equals one, the BR or CALL will be executed. If the status is zero the instruction following the BR or CALL will be executed.

BR (Branch)

A successful BR causes the PC to be loaded from the last six bits I (W) of the BR instruction. If the call latch (CL) is zero, the PA will also be loaded from the PB; however, if the CL is one, the PA will not be altered.

A load page (LDP) instruction can be used prior to a BR to cause program control to be transferred anywhere within the ROM.

Note: A BR within a subroutine CALL is limited to a short branch within the same page in order to preserve the subroutine return address.

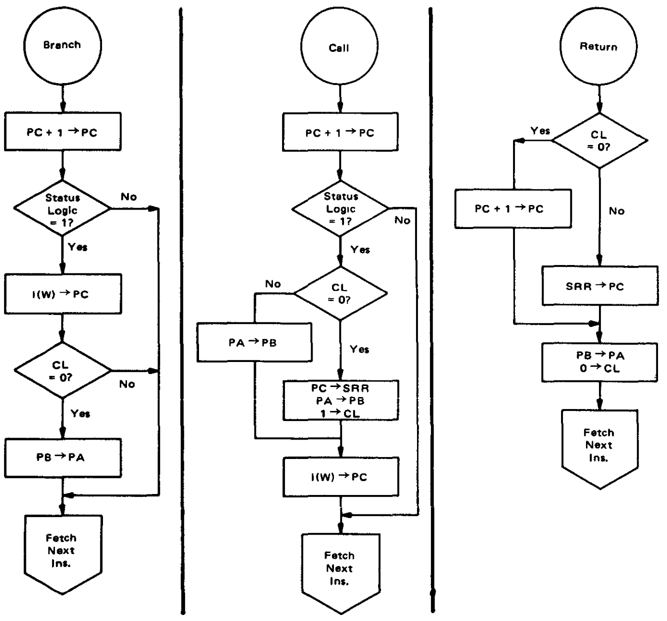


Figure 6 — Internal BR, CALL and RETN Operations

Instruction	Status Logic	Call Latch	Action
BR (Branch)	1	0	I(W) → PC, PB → PA
	1	1	I(W) → PC
	0	*0	PC + 1 → PC, 1 → Status
CALL (Call subroutine)	1	0	PC + 1 → SRR, I(W) → PC, PA → PB, 1 → CL
	1	1	I(W) → PC, PA → PB
	0	*	PC + 1 → PC, 1 → Status Logic
RETN (Return from subroutine)	*	1	SRR → PC, PB → PA, 0 → CL
	*	0	PC + 1 → PC, PB → PA

*	=	Don't Care	SRR	=	Subroutine Return Register
PA	=	Page Address Register	CL	=	Call Latch
PB	=	Page Buffer	I(W)	=	6 Least Significant Bits of a Call or BR
PC	=	Program Counter			

Figure 7 — BR, CALL and RETN Summary

CALL

The CALL instruction permits the use of subroutines in MC141000 programs. The successful CALL instruction causes:

1) the PC to be incremented and stored in the subroutine return register (SRR)
2) the PC to be loaded from the six least significant bits of the CALL instruction
3) the call latch (CL) to be set to one
4) the PB to be exchanged with the PA

Since there is a single level of subroutine-return-address storage, nested subroutines are not permitted. A CALL within a subroutine will cause the return PB to be loaded with the PA. Branch instructions beyond the current page are not permitted within a subroutine since the PB is used as the storage register for the subroutine return page address.

RETN

The return from subroutine instruction (RETN) causes the PC to be loaded from the SRR, the PA to be loaded from the PB, and the CL to be reset.

BR and CALL Summary — Figure 7

The conditions imposed on Branch and Call instructions are:

1) they will only be executed when status is set
2) a long BR or CALL off the current page will result if the PB is loaded by an LDP instruction prior to execution of the BR or CALL
3) only branches within the current page are allowed within a subroutine
4) instruction execution requires six clock cycles whether or not the BR or CALL was successful

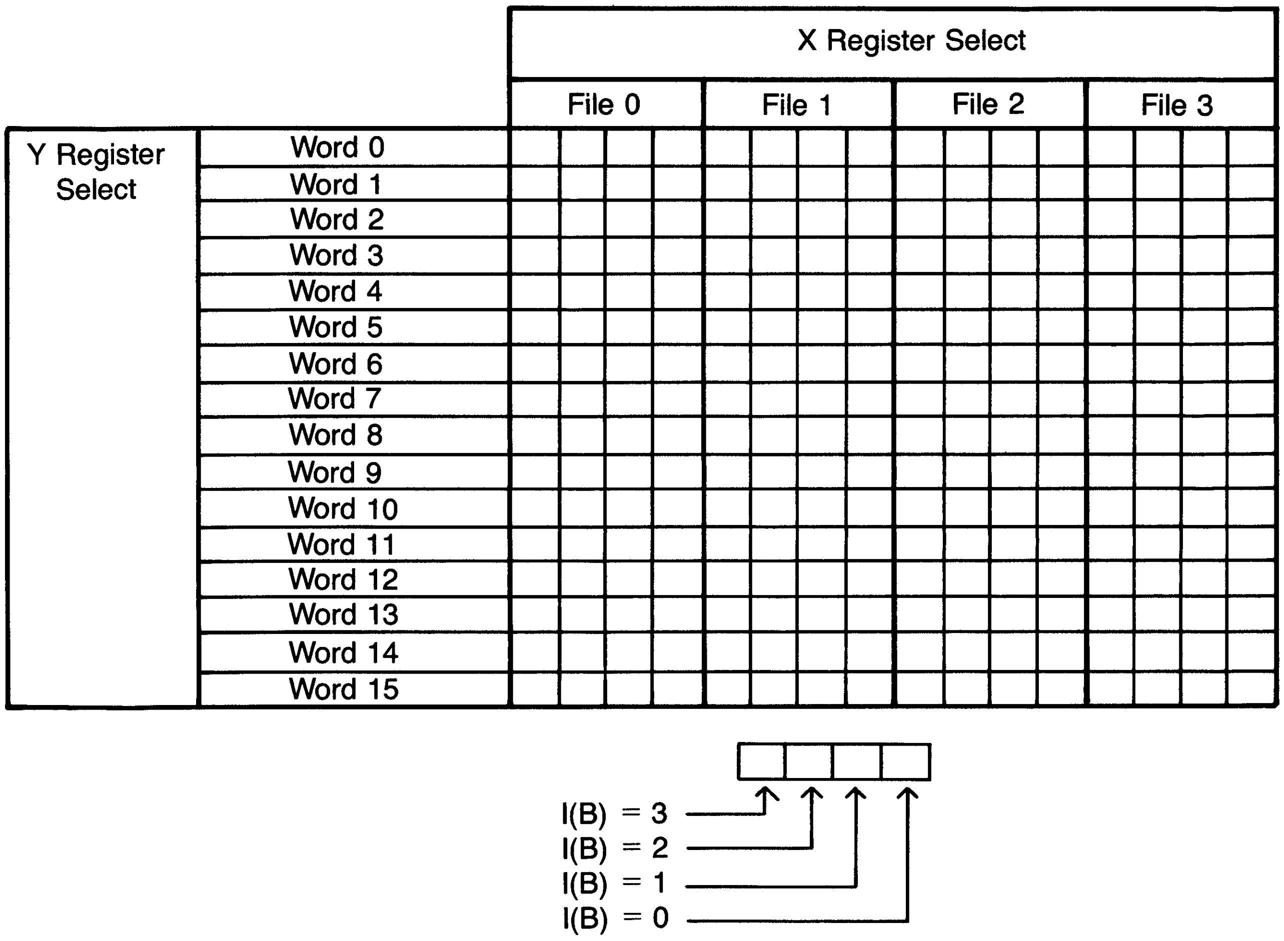


Figure 8. RAM Addressing and Bit Selection

Random Access Memory — RAM

RAM consists of 256 bits organized into 64 4-bit words. For purposes of addressing, the 4-bit words are organized into four files of 16 4-bit words per file, Figure 8.

The X register is decoded to select one of the 4 RAM files; and the Y register is decoded to address one of the 16 words in the selected file.

Instructions which can be used to select the RAM file are the LDX which loads the X register from ROM, and the COMX instruction which complements the X register. The Y register can be loaded from ROM (TCY), from RAM, (TMY), or the accumulator (TAY). The Y register can also be incremented (IYC, TCMIY and TAMIY) and decremented (DYN).

Individual bits within the RAM can be set (SBIT), reset (RBIT), and tested (TBIT1) under program control. The RAM word to be operated on is defined by the X and Y registers, and the 2-bit B field of the bit manipulation instruction selects the bit to be operated on.

Instructions which directly access RAM are:

ALEM —	Accumulator less than or equal to memory
AMAAC —	Add memory to accumulator, store result in accumulator
DMAN —	Load accumulator with decremented memory contents
IMAC —	Load accumulator with incremented memory contents
MNEZ —	Compare for memory not equal to zero
SAMAN —	Subtract the accumulator from the memory and store the result in the accumulator
TAM —	Transfer accumulator to memory
TAMIY —	Transfer accumulator to memory, increment Y register
TAMZA —	Transfer accumulator to memory, load the accumulator with zero
TMA —	Transfer memory to accumulator
TMY —	Transfer memory to Y register
XMA —	Exchange memory and accumulator

Since the Y register is an integral part of the memory addressing scheme, Y register manipulation instructions are important to RAM. The Y register can be changed by the following instructions:

IYC —	Increment the Y register
DYN —	Decrement the Y register
TAY —	Transfer accumulator to Y register
TCY —	Load the Y register with a constant
TMY —	Transfer memory to Y register
TAMIY —	Transfer accumulator to memory, increment to Y register
TCMIY —	Transfer constant to memory, increment Y register

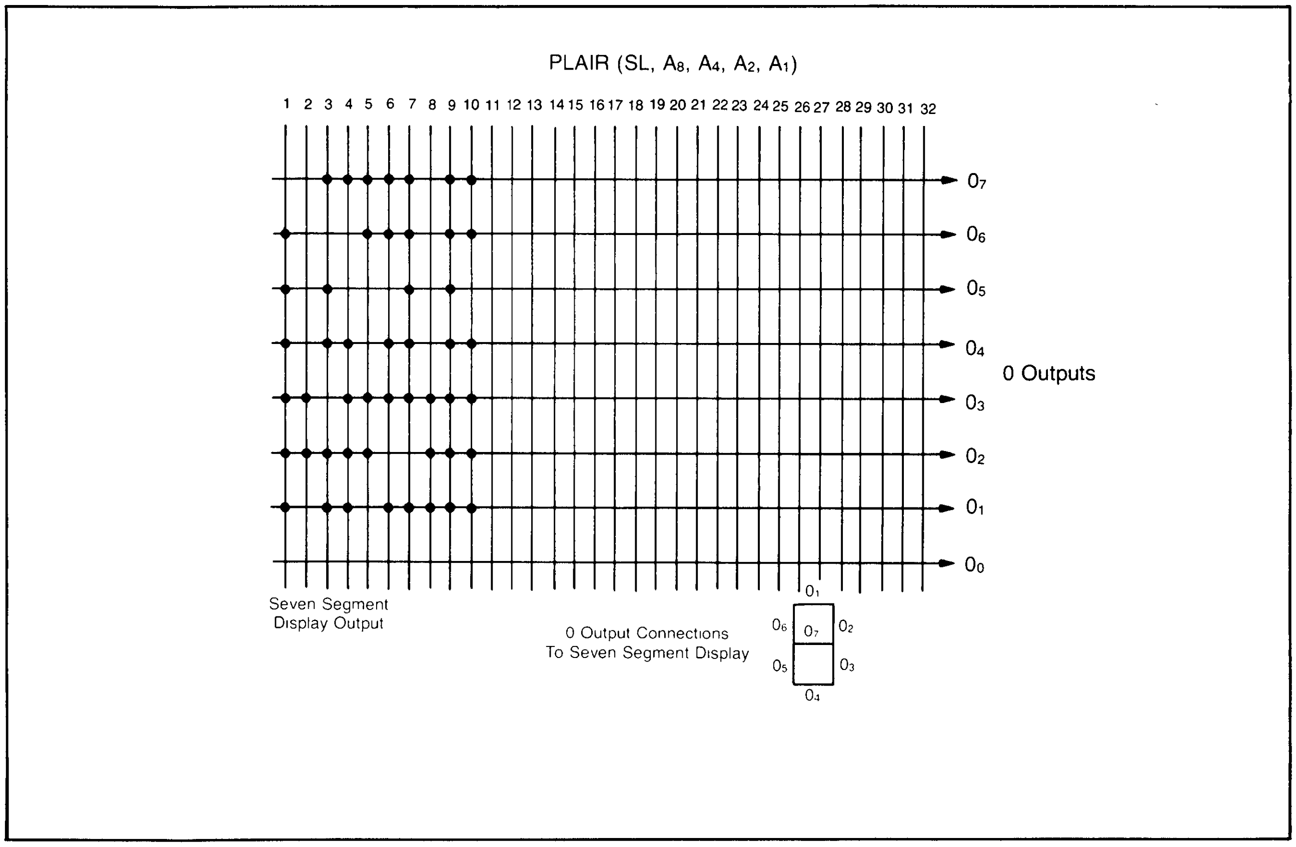


Figure 9 — Output PLA Configured for Seven-Segment Display

Output Ports

Two output ports (R and O) are included in the microcomputer. The MC141000 has 11 R-outputs and the MC14200 has 16 R-outputs while both machines have eight O-outputs. The number of R-outputs is the only difference between the MC141000 and the MC141200.

R-output lines are used primarily as control or "handshake" lines, and to mutiplex external hardware. The R-output which is to be operated on is selected by a binary decode of the contents of the Y register and is set by the SETR instruction and reset RSTR instruction.

The eight O-output lines are the decoded output of the contents of the 5-bit PLAIR. Since the PLAIR is loaded from the A and the SL, these registers must be "set up" prior to an output operation. The status latch can only be loaded by the YNEA (Y register not equal to accumulator) instruction while the contents of the accumulator may be modified by numerous other instructions.

The O output instructions are:

TDO — Transfer data from the accumulator and status latch to the PLAIR
CLO — Clear PLAIR, i.e., load with zeros

In a typical application, the first four R lines might be used as digit selects for outputting a four-digit decimal number using the PLA programmed as a seven-segment decode as shown in Figure 9. The software needed to accomplish this task is shown in the application sections.

Inputs

The input lines consist of the four K-input lines and the initialize (INIT) line. The two input instructions are:

KNEZ — If the K input lines are not equal to zero, set the status logic to one

TKA — Transfer the contents of the four input lines to the accumulator

Operation of the INIT is described previously (Table 1).

Arithmetic Logical Unit (ALU)

The ALU is the calculating and decision-making portion of the MC141000 hardware and consists of a 4-bit adder/comparator and the status logic.

The status logic will be selectively set or reset by add, subtract, increment, decrement, compare and bit-test operations. Other instructions always set the status logic to a one.

The adder/comparator can add, subtract, compare two numbers, add +1, −1, 6, 8, and 10.

The arithmetic instructions are:

AMAAC —	Add memory to accumulator, results to accumulator. Carry to status
SAMAN —	Subtract accumulator from memory, results to accumulator. If no borrow, one to status
IMAC —	Load memory into accumulator, increment accumulator. Carry to status
DMAN —	Load memory into accumulator, decrement accumulator. If no borrow, one to status
IA —	Increment accumulator, no status effect
IYC —	Increment Y register. Carry to status
DAN —	Decrement accumulator. If no borrow, one to status
DYN —	Decrement Y register. If no borrow, one to status
A8AAC —	Add 8 to accumulator, results to accumulator. Carry to status
A10AAC —	Add 10 to accumulator, results to accumulator. Carry to status
A6AAC —	Add 6 to accumulator, results to accumulator. Carry to status
CPAIZ —	Complement accumulator increment. If zero, one to status

The logical instructions are:

ALEM —	If accumulator less than or equal to memory, one to status
ALEC —	If accumulator less than or equal to a constant, one to status
KNEZ —	If K-inputs not all zero, one to status
MNEZ —	If memory not equal to zero, one to status
TBIT1 —	If the selected bit is one, one to status
YNEA —	If Y register not equal to accumulator, one to status and status latch
YNEC —	If Y register not equal to a constant, one to status

Instruction Decode

The instruction decode logic latches every instruction fetched from ROM and configures the internal logic to correctly execute the curthe current instruction. The MC141000 includes within the instruction decode logic the capability of modifying the standard instruction set. Typical examples of useful nonstandard instructions are:

SRDY —	Set R-Output and decrement Y
TDOIY —	Transfer A and SL to PLAIR and increment Y
TKM —	Transfer K inputs to memory and increment the Y Register
ANEM —	A not equal to M(X, Y)

The factory should be consulted for feasibility of specific instruction-set modifications.

INSTRUCTION DETAILS

On the following pages, the MC141000 Instruction Set is described in detail. The descriptions use the following format:

MNEMONIC

INSTRUCTION DEFINITION —

ACTION:	Symbolically shows the instructions effect. One or more cases may be shown here, depending on the instruction.
DESCRIPTION:	Explains the instruction operation in detail.
STATUS:	Shows the effect of the instruction on the status logic.
OPERATION:	A diagramatic representation of the effect of the operation on pertinent registers. Register contents BEFORE execution and AFTER execution are shown. Any register not shown (with the exception of the PC) is unchanged by the instruction. Where two conditions may result from execution of an instruction, a diagonal line separates examples showing both results.
OPERATION CODE:	The binary code, including field designations.

ALEC

IF ACCUMULATOR IS LESS THAN OR EQUAL TO CONSTANT, ONE TO STATUS.

ACTION:

A < I(C)	A = I(C)	A > I(C)
1 → S	1 → S	0 → S

DESCRIPTION: If the contents of the accumulator are less than or equal to the C field of the instruction, the status will be set to one; if not, the status is set to zero. If the constant 15 is used, this instruction effectively becomes a no-op.

STATUS: Set if A ≤ constant, reset if not.

OPERATION:

	Registers		
	A	I(C)	S
BEFORE	4	9 / 2	*
AFTER	4	9 / 2	1 / 0

OPERATION CODE:

0	1	1	1	C LSB … MSB

ALEM

IF ACCUMULATOR IS LESS THAN OR EQUAL TO MEMORY, ONE TO STATUS.

ACTION:

A < M(X, Y)	A = M(X, Y)	A > M(X, Y)
1 → S	1 → S	0 → S

DESCRIPTION: If the contents of the accumulator are less than or equal to the contents of RAM file X, word Y status is set, if not status is reset.

STATUS: Set if A ≤ RAM contents, reset if not.

OPERATION:

	Registers		
	A	M(X, Y)	S
BEFORE	6	10 / 4	*
AFTER	6	10 / 4	1 / 0

OPERATION CODE:

0	0	1	0	1	0	0	1

AMAAC

ADD MEMORY TO ACCUMULATOR. LOAD RESULT INTO ACCUMULATOR.

ACTION: A + M(X, Y) → A
CARRY → S

DESCRIPTION: The contents of RAM file X, word Y are added to the contents of the accumulator, and the result is loaded into the accumulator. The status is set if a carry results from the addition and reset if there is no carry.

STATUS: Set if carry, reset if not.

OPERATION:

	Registers		
	A	M(X, Y)	S
BEFORE	4 / 9	8	*
AFTER	12 / 1	8	0 / 1

OPERATION CODE:

0	0	1	0	0	1	0	1

*Don't Care

A6AAC

ADD SIX TO ACCUMULATOR.

ACTION: A + 6 → A
CARRY → S

DESCRIPTION: Six is added to the accumulator. If a carry results, the status is set. This instruction is commonly used for correction during BCD addition.

STATUS: Set on carry, reset if not.

OPERATION:

	Registers	
	A	S
BEFORE	4 / 15	*
AFTER	10 / 5	0 / 1

OPERATION CODE:

0	0	0	0	0	1	1	0

A8AAC

ADD EIGHT TO ACCUMULATOR.

ACTION: A + 8 → A
CARRY → S

DESCRIPTION: Add 8 to the accumulator. Set the status if a carry results.

STATUS: Set on carry, reset if not.

OPERATION:

	Registers	
	A	S
BEFORE	6 / 9	*
AFTER	14 / 1	0 / 1

OPERATION CODE:

0	0	0	0	0	0	0	1

A10AAC

ADD TEN TO ACCUMULATOR.

ACTION: A + 10 → A
CARRY → S

DESCRIPTION: Ten is added to the accumulator. If a carry results, the status is set. This instruction is commonly used for correction during BCD subtraction.

STATUS: Set on carry, reset if not.

OPERATION:

	Registers	
	A	S
BEFORE	3 / 9	*
AFTER	13 / 4	0 / 1

OPERATION CODE:

0	0	0	0	0	1	0	1

*Don't care

BR

BRANCH ON STATUS EQUALS ONE.

ACTION:

	BRANCH		NO BRANCH

If	S = 1
	CL = 0
		S = 1
		CL = 1
			S = 0
			CL = Don't Care

Then	I(W) → PC PB → PA	I(W) → PC	PC + 1 → PC

DESCRIPTION: If the status is zero, the next sequential instruction after the branch will be executed and the program will not branch. If the status is set and the call latch is one, the program counter is loaded from the W field of the instruction and the program will branch within the page. Interpage branching is not successful if CL = 1. If the status is set and the call latch is zero, the page address will be loaded from the page buffer and the program counter will be loaded from the instruction, permitting a branch to anywhere in ROM. Execution of the BR instruction requires 6 clock cycles whether or not the branch is successful.

STATUS: Set.

OPERATION CODE:

1	0	W
		MSB — LSB

CALL

CALL SUBROUTINE ON STATUS EQUALS ONE.

ACTION:

	CALL		NO CALL

If	S = 1
	CL = 0
		S = 1
		CL = 1
			S = 0

Then	I(W) → PC PC → SRR PB ↔ PA 1 → CL	I(W) → PC PA ↔ PB	PC + 1 → PC

DESCRIPTION: If the status is zero, the call is unsuccessful and the next sequential instruction after the call is executed. If the status is set and the call latch is one, the program counter is loaded from the W field of the instruction and the program will branch within the page and the return page may be lost. If the status is set and the call latch is zero, the page address and the page buffer are exchanged and the program counter is incremented, stored in the subroutine return register, and loaded from the W field of the instruction, permitting a branch to anywhere in ROM.

STATUS: Set.

OPERATION CODE:

1	1	W
		MSB — LSB

CLA

CLEAR ACCUMULATOR.

ACTION: $0 \rightarrow A$

DESCRIPTION: The accumulator is loaded with zero.

STATUS: Set.

OPERATION:

	Registers	
	A	S
BEFORE	7	*
AFTER	0	1

OPERATION CODE:

0	0	1	0	1	1	1	1

CLO

CLEAR PLAIR OUTPUT.

ACTION: $0 \rightarrow$ PLAIR

DESCRIPTION: The five bits of the programmable logic array input register are set to zero.

STATUS: Set.

OPERATION:

	Registers
	PLAIR
BEFORE	*
AFTER	0

OPERATION CODE:

0	0	0	0	1	0	1	1

COMX

COMPLEMENT X REGISTER.

ACTION: $\overline{X} \rightarrow X$

DESCRIPTION: The contents of the X register are one's complemented.

STATUS: Set.

OPERATION:

	Registers	
	X	S
BEFORE	0 / 2	*
AFTER	3 / 1	1

OPERATION CODE:

0	0	0	0	0	0	0	0

*Don't care

CPAIZ

COMPLEMENT AND INCREMENT ACCUMULATOR.

ACTION: $\overline{A} + 1 \rightarrow A$
Carry $\rightarrow$ S

DESCRIPTION: The accumulator is complemented and incremented. If a carry results status is set, no carry resets status.

STATUS: Set on carry, reset if not.

OPERATION:

	Registers	
	A	S
BEFORE	3 / 0	*
AFTER	13 / 0	0 / 1

OPERATION CODE:

0	0	1	0	1	1	0	1

DAN

DECREMENT ACCUMULATOR.

ACTION: $A - 1 \rightarrow A$
$1 -$ Borrow $\rightarrow$ S

DESCRIPTION: The contents of the accumulator are decremented by one. If no borrow results, the status is set to one. If the accumulator contains zero prior to execution, status is reset and the accumulator is set to 15.

STATUS: Set on no borrow, reset on borrow.

OPERATION:

	Registers	
	A	S
BEFORE	9 / 0	*
AFTER	8 / 15	1 / 0

OPERATION CODE:

0	0	0	0	0	1	1	1

DMAN

DECREMENTED MEMORY INTO ACCUMULATOR.

ACTION: $M(X, Y) - 1 \rightarrow A$
$1 -$ Borrow $\rightarrow$ S

DESCRIPTION: The contents of M(X, Y) are loaded into the accumulator and decremented. The status is set only if M(X, Y) = 0, A = F. M(X, Y) is not modified by this operation.

STATUS: Set if no borrow, reset if borrow.

OPERATION:

	Registers		
	A	M(X, Y)	S
BEFORE	*	4 / 0	*
AFTER	3 / 15	4 / 0	1 / 0

OPERATION CODE:

0	0	1	0	1	0	1	0

*Don't care

DYN

DECREMENT Y REGISTER.

ACTION: $Y - 1 \rightarrow Y$
$1 - \text{Borrow} \rightarrow \text{Status}$

DESCRIPTION: The Y register is decremented by 1. If no borrow results, the status bit is set. If Y = 0 prior to execution, a borrow will result and the status logic will be reset. All other values of Y will result in status being set.

STATUS: Set on no borrow, reset on borrow.

OPERATION:

	Registers	
	Y	S
BEFORE	6 / 0	*
AFTER	5 / 15	1 / 0

OPERATION CODE:

0	0	1	0	1	1	0	0

IA

INCREMENT ACCUMULATOR.

ACTION: $A + 1 \rightarrow A$

DESCRIPTION: The contents of the accumulator are incremented by one.

STATUS: Set.

OPERATION:

	Registers	
	A	S
BEFORE	5 / 15	*
AFTER	6 / 0	1

OPERATION CODE:

0	0	0	0	1	1	1	0

IMAC

INCREMENTED MEMORY TO ACCUMULATOR.

ACTION: $M(X, Y) + 1 \rightarrow A$
$\text{CARRY} \rightarrow S$

DESCRIPTION: The contents of RAM file X, word Y are loaded into the accumulator and incremented. The status bit is set to one on a carry or to zero for no carry.

STATUS: Set if carry, reset if no carry.

OPERATION:

	Registers		
	A	M(X, Y)	S
BEFORE	*	2 / 15	*
AFTER	3 / 0	2 / 15	0 / 1

OPERATION CODE:

0	0	1	0	1	0	0	0

*Don't care

IYC

INCREMENT Y REGISTER.

ACTION: $Y + 1 \rightarrow Y$
$\text{CARRY} \rightarrow S$

DESCRIPTION: The contents of the Y register are incremented by one. If a carry results, the status bit is set to one.

STATUS: Set on carry, reset if no carry.

OPERATION:

	Registers	
	Y	S
BEFORE	4 / 15	*
AFTER	5 / 0	0 / 1

OPERATION CODE:

0	0	1	0	1	0	1	1

KNEZ

IF K INPUTS ARE NOT EQUAL TO ZERO, SET STATUS.

ACTION: $K = 0$ $\quad$ $K \neq 0$
$0 \rightarrow S$ $\quad$ $1 \rightarrow S$

DESCRIPTION: Compare the data on the four K input lines with zero. If the input data is not zero, set the status bit.

STATUS: Set if $K \neq 0$, reset otherwise.

OPERATION:

	Registers	
	K	S
BEFORE	7 / 0	*
AFTER	7 / 0	1 / 0

OPERATION CODE:

0	0	0	0	1	0	0	1

LDP

LOAD THE PAGE BUFFER.

ACTION: $I(C) \rightarrow PB$

DESCRIPTION: The C field of the instruction is loaded into the page buffer register.

STATUS: Set

OPERATION:

	Registers	
	I(C)	PB
BEFORE	4	*
AFTER	*	4

OPERATION CODE:

0	0	0	1	C			
				LSB			MSB

*Don't care

LDX

LOAD X REGISTER WITH A CONSTANT.

ACTION: I(B) → X

DESCRIPTION: The B field of the instruction is loaded into the X register.

STATUS: Set.

OPERATION:

	Registers		
	X	I(B)	S
BEFORE	*	2	*
AFTER	2	*	1

OPERATION CODE:

0	0	1	1	1	1	B LSB | MSB

MNEZ

IF MEMORY NOT EQUAL TO ZERO, ONE TO STATUS.

ACTION:

M(X,Y) = 0	M(X,Y) ≠ 0
0 → S	1 → S

DESCRIPTION: If the contents of memory file X, word Y are equal to zero, the status logic is reset. If the contents of that memory location are not zero, the status is set.

STATUS: Reset if M(X, Y) = 0, set if M(X, Y) ≠ 0.

OPERATION:

	Registers	
	M(X,Y)	S
BEFORE	**0 / 7**	*
AFTER	**0 / 7**	**0 / 1**

OPERATION CODE:

0	0	1	0	0	1	1	0

RBIT

RESET RAM BIT.

ACTION: 0 → M(X,Y, B)

DESCRIPTION: The bit defined by the I(B) field of RAM file X, word Y, is reset to zero. RAM bits are

3	2	1	0
MSB			LSB

STATUS: Set.

OPERATION:

	Registers		
	M(X,Y)	S	I(B)
BEFORE	**12 / 6**	*	3
AFTER	**4 / 6**	1	*

OPERATION CODE:

0	0	1	1	0	1	B LSB | MSB

*Don't care

RETN

RETURN FROM SUBROUTINE.

ACTION:

IF	CL = 1	CL = 0
THEN	SRR → PC PB → PA 0 → CL	PC + 1 → PC PB → PA

DESCRIPTION: The RETN instruction loads the program counter from the subroutine return register, loads the page address register from the page buffer and resets the call latch.

STATUS: Set.

OPERATION CODE:

0	0	0	0	1	1	1	1

RSTR

RESET R OUTPUT LINE.

ACTION: 0 → R(Y)

DESCRIPTION: The R output line selected by the Y register is reset to zero.

STATUS: Set.

OPERATION:

	Registers	
	R(Y)	S
BEFORE	*	*
AFTER	0	1

OPERATION CODE:

0	0	0	0	1	1	0	0

SAMAN

SUBTRACT ACCUMULATOR FROM MEMORY. LOAD RESULT INTO ACCUMULATOR.

ACTION: M(X,Y) − A → A
1 − BORROW → S

DESCRIPTION: The contents of the accumulator are subtracted from the contents of RAM file X, word Y, and the result is loaded into the accumulator. If A is greater than M(X,Y), the status bit is reset to zero, but if A is less than or equal to M(X,Y), the status bit is set to one.

STATUS: Set if no borrow, reset if borrow.

OPERATION:

	Registers		
	A	M(X,Y)	S
BEFORE	**3 / 6**	4	*
AFTER	**1 / 14**	4	**1 / 0**

OPERATION CODE:

0	0	1	0	0	1	1	1

*Don't care

SBIT

SET RAM BIT.

ACTION: $1 \rightarrow M(X,Y,B)$

DESCRIPTION: The bit defined by the I(B) field of RAM file X, word Y is set to a one. RAM bits are

3	2	1	0
MSB			LSB

STATUS: Set.

OPERATION:

	Registers		
	M(X,Y)	I(B)	S
BEFORE	10 / 14	2	*
AFTER	14 / 14	*	1

OPERATION CODE:

0	0	1	1	0	0	B	
						LSB	MSB

SETR

SET R OUTPUT LINE.

ACTION: $1 \rightarrow R(Y)$

DESCRIPTION: The R output line selected by the Y register is set to one.

STATUS: Set.

OPERATION:

	Registers	
	R(Y)	S
BEFORE	*	*
AFTER	1	1

OPERATION CODE:

0	0	0	0	1	1	0	1

TAM

TRANSFER ACCUMULATOR TO MEMORY.

ACTION: $A \rightarrow M(X,Y)$

DESCRIPTION: The contents of the accumulator are stored in RAM file X, word Y.

STATUS: Set.

OPERATION:

	Registers		
	A	M(X,Y)	S
BEFORE	6	*	*
AFTER	6	6	1

OPERATION CODE:

0	0	0	0	0	0	1	1

*Don't care

TAMIY

TRANSFER ACCUMULATOR TO MEMORY. INCREMENT Y REGISTER.

ACTION: $A \rightarrow M(X,Y),\ Y + 1 \rightarrow Y$

DESCRIPTION: The contents of the accumulator are stored in RAM file X, word Y. The contents of the Y register are incremented.

STATUS: Set.

OPERATION:

	Registers			
	Y	A	M(X,Y)	S
BEFORE	5	3	*	*
AFTER	6	3	3	1

OPERATION CODE:

0	0	1	0	0	0	0	0

TAMZA

TRANSFER ACCUMULATOR TO MEMORY. ZERO ACCUMULATOR.

ACTION: $A \rightarrow M(X,Y),\ 0 \rightarrow A$

DESCRIPTION: The contents of the accumulator are stored in RAM file X, word Y. The accumulator is then loaded with zero.

STATUS: Set.

OPERATION:

	Registers		
	A	M(X,Y)	S
BEFORE	8	*	*
AFTER	0	8	1

OPERATION CODE:

0	0	0	0	0	1	0	0

TAY

TRANSFER ACCUMULATOR TO Y REGISTER.

ACTION: $A \rightarrow Y$

DESCRIPTION: The contents of the accumulator are loaded into the Y register.

STATUS: Set.

OPERATION:

	Registers		
	Y	A	S
BEFORE	*	9	*
AFTER	9	9	1

OPERATION CODE:

0	0	1	0	0	1	0	0

*Don't care

TBIT1

TRANSFER ONE RAM BIT TO STATUS

ACTION: M(X,Y, B) → S

DESCRIPTION: If the bit selected by the I(B) field in RAM file X, word Y is a one, the status logic is set to one. If it is zero, the status logic is reset to zero. RAM bits are

3	2	1	0
MSB			LSB

STATUS: Set if bit = 1, reset if bit = 0.

OPERATION:

	Registers		
	M(X,Y)	B	S
BEFORE	9 / 14	0	*
AFTER	9 / 14	*	1 / 0

OPERATION CODE:

0	0	1	1	1	0	B	
						LSB	MSB

TCMIY

TRANSFER CONSTANT TO MEMORY. INCREMENT Y REGISTER.

ACTION: I(C) → M(X,Y)
Y + I → Y

DESCRIPTION: The ROM constant contained in the C field of the instruction, I(C), is loaded into RAM file X, word Y. The Y register is incremented by one.

STATUS: Set.

OPERATION:

	Registers			
	Y	M(X,Y)	S	I(C)
BEFORE	4	*	*	9
AFTER	5	9	1	*

OPERATION CODE:

0	1	1	0	C			
				LSB			MSB

TCY

TRANSFER CONSTANT TO Y REGISTER.

ACTION: I(C) → Y

DESCRIPTION: The constant contained in the C field of the instruction, I(C), is loaded into the Y register.

STATUS: Set.

OPERATION:

	Registers		
	Y	S	I(C)
BEFORE	*	*	10
AFTER	10	1	*

OPERATION CODE:

0	1	0	0	C			
				LSB			MSB

*Don't care

TDO

TRANSFER DATA TO OUTPUTS

ACTION: SL,A → PLAIR

DESCRIPTION: The status latch and accumulator contents are loaded into the programmable logic array input register. The 5-bit word selecting one of the 32 possible output terms is:

SL	A_8	A_4	A_2	A_1
MSB				LSB

STATUS: Set.

OPERATION:

	Registers			
	SL	A	PLAIR	S
BEFORE	1 / 0	4 / 7	*/*	*
AFTER	1 / 0	4 / 7	36 / 7	1

OPERATION CODE:

0	0	0	0	1	0	1	0

TKA

TRANSFER K INPUTS INTO ACCUMULATOR.

ACTION: K → A

DESCRIPTION: Data from the four K inputs is loaded into the accumulator.

STATUS: Set.

OPERATION:

	Registers		
	A	S	K
BEFORE	*	*	6
AFTER	6	1	*

OPERATION CODE:

0	0	0	0	1	0	0	0

TMA

TRANSFER MEMORY TO ACCUMULATOR.

ACTION: M(X,Y) → A

DESCRIPTION: The contents of RAM file X, word Y, are loaded into the accumulator.

STATUS: Set.

OPERATION:

	Registers		
	A	M(X,Y)	S
BEFORE	*	4	*
AFTER	4	4	1

OPERATION CODE:

0	0	1	0	0	0	0	1

*Don't care

TMY

TRANSFER MEMORY TO Y REGISTER.

ACTION: M(X,Y) → Y

DESCRIPTION: The contents of RAM file X, word Y, are transferred to the Y register.

STATUS: Set.

OPERATION:

	Registers		
	Y	M(X,Y)	S
BEFORE	*	11	*
AFTER	11	11	1

OPERATION CODE:

0	0	1	0	0	0	1	0

TYA

TRANSFER Y REGISTER TO ACCUMULATOR.

ACTION: Y → A

DESCRIPTION: The contents of the Y register are loaded into the accumulator.

STATUS: Set.

OPERATION:

	Registers		
	Y	A	S
BEFORE	4	*	*
AFTER	4	4	1

OPERATION CODE:

0	0	1	0	0	0	1	1

XMA

EXCHANGE MEMORY AND ACCUMULATOR.

ACTION: M(X,Y) ←→ A

DESCRIPTION: The contents of RAM file X, word Y and the accumulator are exchanged.

STATUS: Set.

OPERATION:

	Registers		
	A	M(X,Y)	S
BEFORE	11	5	*
AFTER	5	11	1

OPERATION CODE:

0	0	1	0	1	1	1	0

*Don't care

YNEA

IF Y REGISTER ≠ A, ONE TO STATUS AND STATUS LATCH.

ACTION:	Y ≠ A 1 → S 1 → SL	Y = A 0 → S 0 → SL

DESCRIPTION: The contents of the accumulator and the Y register are compared. If they are not equal, the status and status latch are set. This is the only instruction which sets or resets the status latch, consequently if SL = 0, PLA terms 0–15 may be output, if SL = 1, PLA terms 16–31 may be output.

STATUS: Set if Y ≠ A, reset otherwise.

OPERATION:

	Registers			
	Y	A	S	SL
BEFORE	12	8 / 12	*	*
AFTER	12	8 / 12	1 / 0	1 / 0

OPERATION CODE:

0	0	0	0	0	0	1	0

YNEC

IF Y REGISTER ≠ CONSTANT, ONE TO STATUS.

ACTION:	Y = I(C) 0 → S	Y ≠ I(C) 1 → S

DESCRIPTION: The contents of the Y register are compared with the C field of the instruction. If they are equal, the status logic is reset to zero, if they are not equal, the status logic is set to one.

STATUS: Set if Y ≠ C, reset if Y = C.

OPERATION:

	Registers		
	I(C)	Y	S
BEFORE	4	4 / 3	*
AFTER	4	4 / 3	0 / 1

OPERATION CODE:

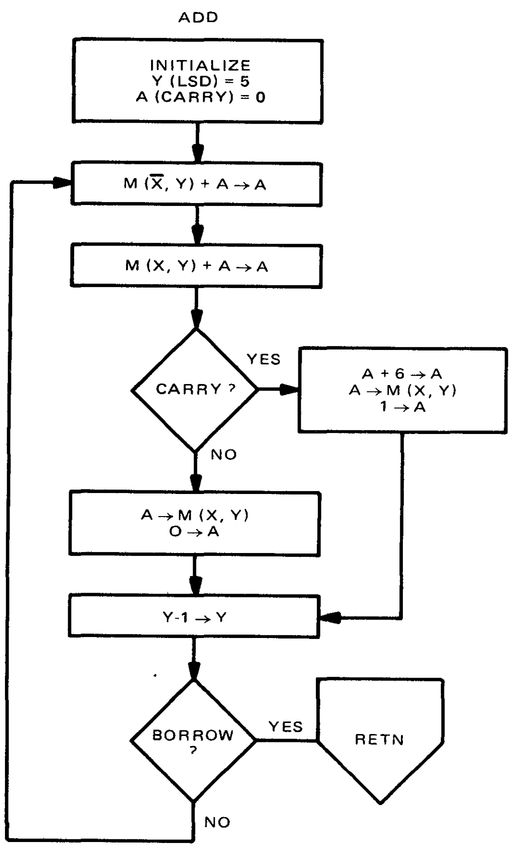
					C		
0	1	0	1	LSB			MSB

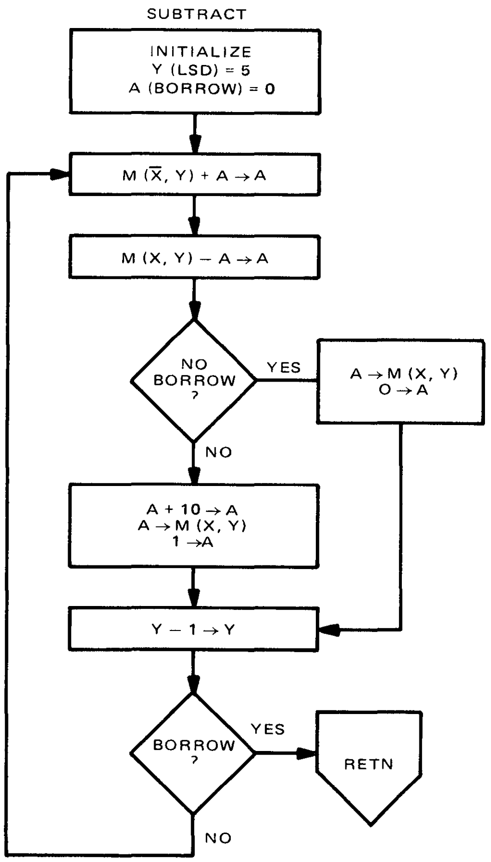
*Don't care

APPLICATIONS AND SOFTWARE EXAMPLES

The following hardware and software examples illustrate how the MC141000 is used in typical applications. The examples include BCD addition and subtraction, controlling a display, monitoring a keypad, interfacing an external CMOS memory, and expanding the number of R-Outputs.

BCD Addition and Subtraction Flowchart





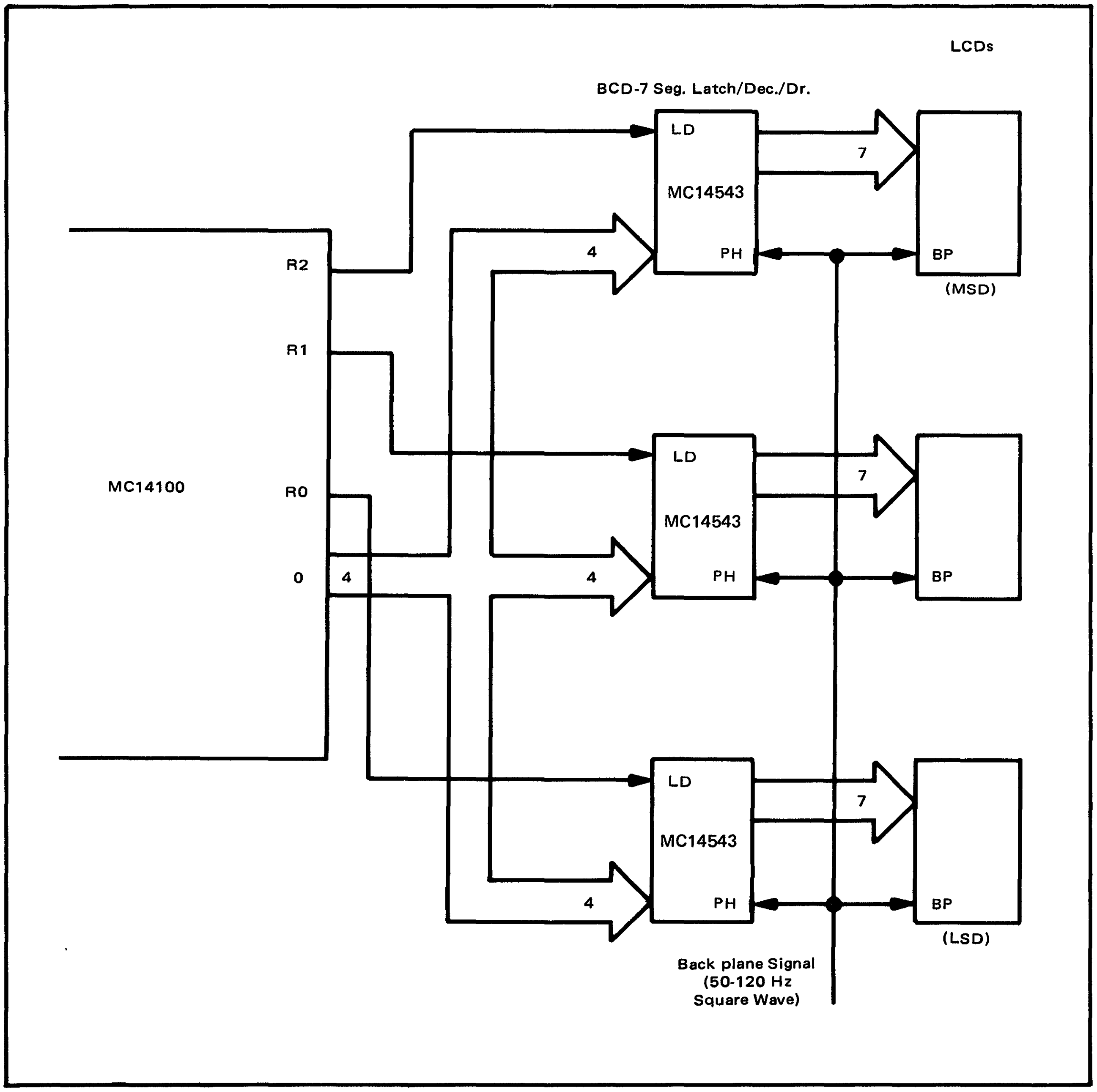
```
*******************************************************************************
BCD ADDITION SUBROUTINE SUMS TWO 6-DIGIT BASE TEN
NUMBERS LOCATED IN Y=0(MSD) THRU Y=5(LSD) OF COMPLEMENTARY
X FILES. THE AUGEND IN X IS REPLACED BY THE SUM AND THE ADDEND IN
X-COMPLEMENT IS UNCHANGED. AN OVERFLOW CONDITION IS
INDICATED BY A CARRY=1 IN THE ACCUMULATOR ON RETURN.
*******************************************************************************

ADD     CLA             CLEAR CARRY
        TCY     5       ADDRESS LSD
ADD1    COMX            COMPLEMENT X (ADDRESS ADDEND)
        AMAAC           ADDEND DIGIT + CARRY
        COMX            ADDRESS AUGEND
        AMAAC           ADD AUGEND TO ACCUMULATOR
        BR      CARRY   BRANCH IF SUM CAUSES CARRY
        ALEC    9       VALID NUMBER ?
        BR      ADDOK   YES, BRANCH
CARRY   A6AAC           SUM GREATER THAN 10, ADD CORRECTION
        TAMZA           UNITS TO MEMORY, ZERO ACCUMULATOR
        IA              CARRY = 1 TO ACCUMULATOR
ADD2    DYN             DECREMENT Y UNTIL BORROW
        BR      ADD1    BRANCH ON NO BORROW
        RETN            RETURN ON BORROW
*
ADDOK   TAMZA           SUM TO MEMORY, ZERO CARRY
        BR      ADD2    BRANCH TO ADDRESS NEXT DIGIT
```

```
*******************************************************************************
BCD SUBTRACTION SUBROUTINE SUBTRACTS TWO 6-DIGIT BASE TEN
NUMBERS LOCATED IN Y=0 (MSD) THRU Y=5 (LSD) OF
COMPLEMENTARY X FILES. THE MINUEND IN X IS REPLACED BY THE
DIFFERENCE AND THE SUBTRAHEND IN X-COMPLEMENT IS UNCHANGED.
AN UNDERFLOW CONDITION IS INDICATED BY A BORROW IN THE
ACCUMULATOR ON RETURN.
*******************************************************************************

SUBT    TCY     5       ADDRESS LSD AT Y=5
        CLA             CLEAR BORROW
SUBT1   COMX            COMPLEMENT X REGISTER
        AMAAC           ADD SUBTRAHEND DIGIT TO ACCUMULATOR
        COMX            COMPLEMENT X
        SAMAN           SUBTRACT SUBTRAHEND FROM MINUEND
        BR      SUBOK   BRANCH IF NO BORROW
        A10AAC          ADD CORRECTION FACTOR IF BORROW
        TAMZA           UNITS TO MEMORY, ACCUMULATOR = 0
        IA              SET BORROW = 1
SUBT2   DYN             DECREMENT Y TO ADDRESS NEXT DIGIT
        BR      SUBT1   BRANCH IF NO BORROW
        RETN            RETURN TO CALLING ADDRESS
*
SUBOK   TAMZA           DIFFERENCE TO MEMORY, BORROW = 0
        BR      SUBT2   BRANCH TO ADDRESS NEXT DIGIT
```

LCD Display Hardware and Software



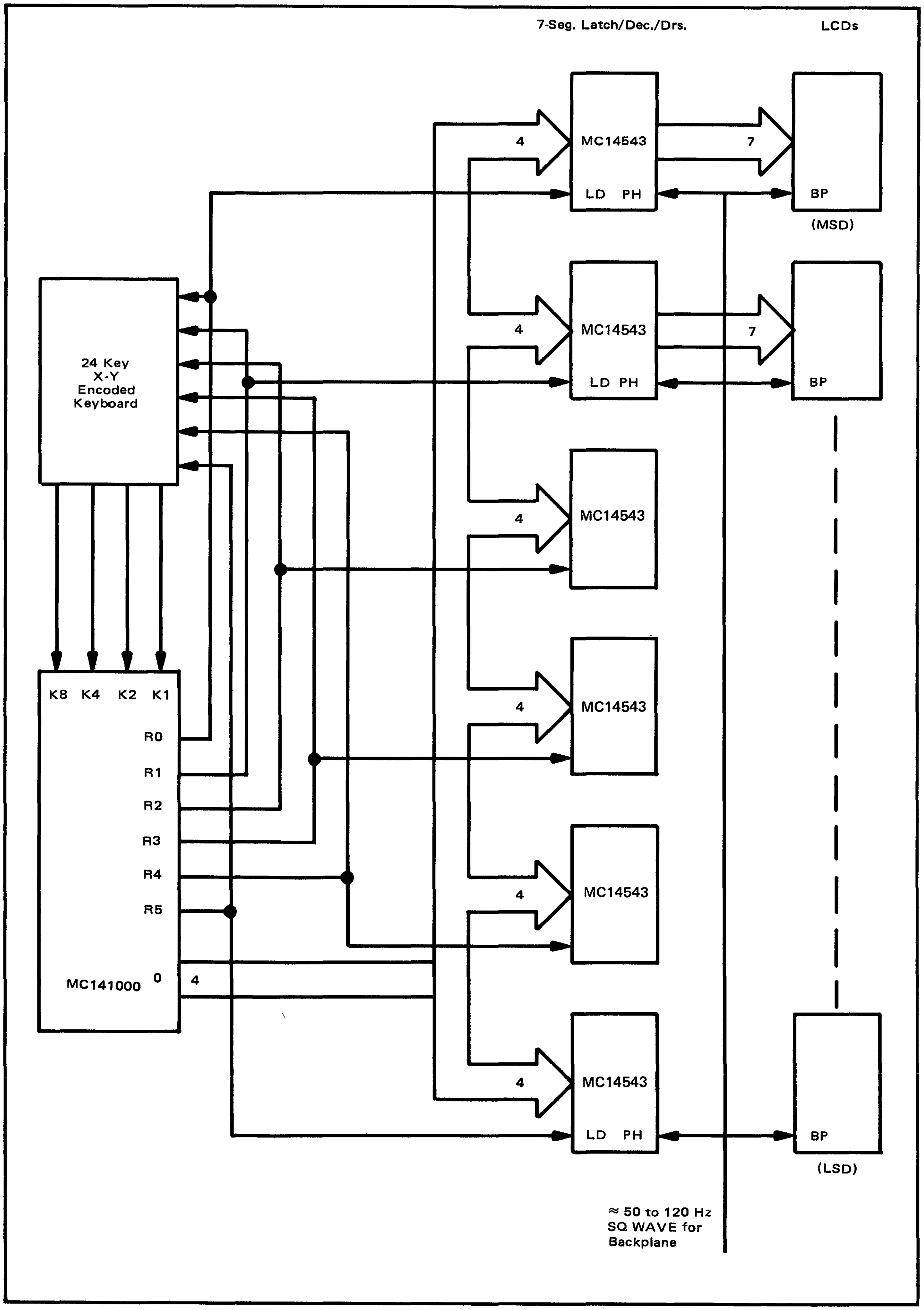
LCDs
BCD-7 Seg. Latch/Dec./Dr.
LD
MC14543
PH
BP
(MSD)
7
4
R2
R1
R0
0
MC14100
(LSD)
Back plane Signal (50-120 Hz Square Wave)

```
*******************************************************************************
SUBROUTINE TO SCAN AN LCD DISPLAY. THREE DIGITS ARE ASSUMED
HERE, BUT UP TO SIXTEEN CAN BE ACCOMMODATED MERELY BY
CHANGING THE CONSTANT ON THE TCY INSTRUCTION.

THE DATA IS ASSUMED TO BE IN RAM FILE 0 WITH Y=0 BEING THE LSD.

SCAN   LDX    0       ADDRESS THE DATA
       TCY    2
NEXT   TMA            TRANSFER THE DATA TO
       TDO              THE 0-OUTPUTS
       SETR           LATCH THE DATA INTO
       RSTR             THE LCD DRIVER
       DYN            SELECT NEXT DIGIT, OR
       BR     NEXT      IF DONE, RETURN
       RETN
```

LCD Display Plus Keyboard Hardware



Software

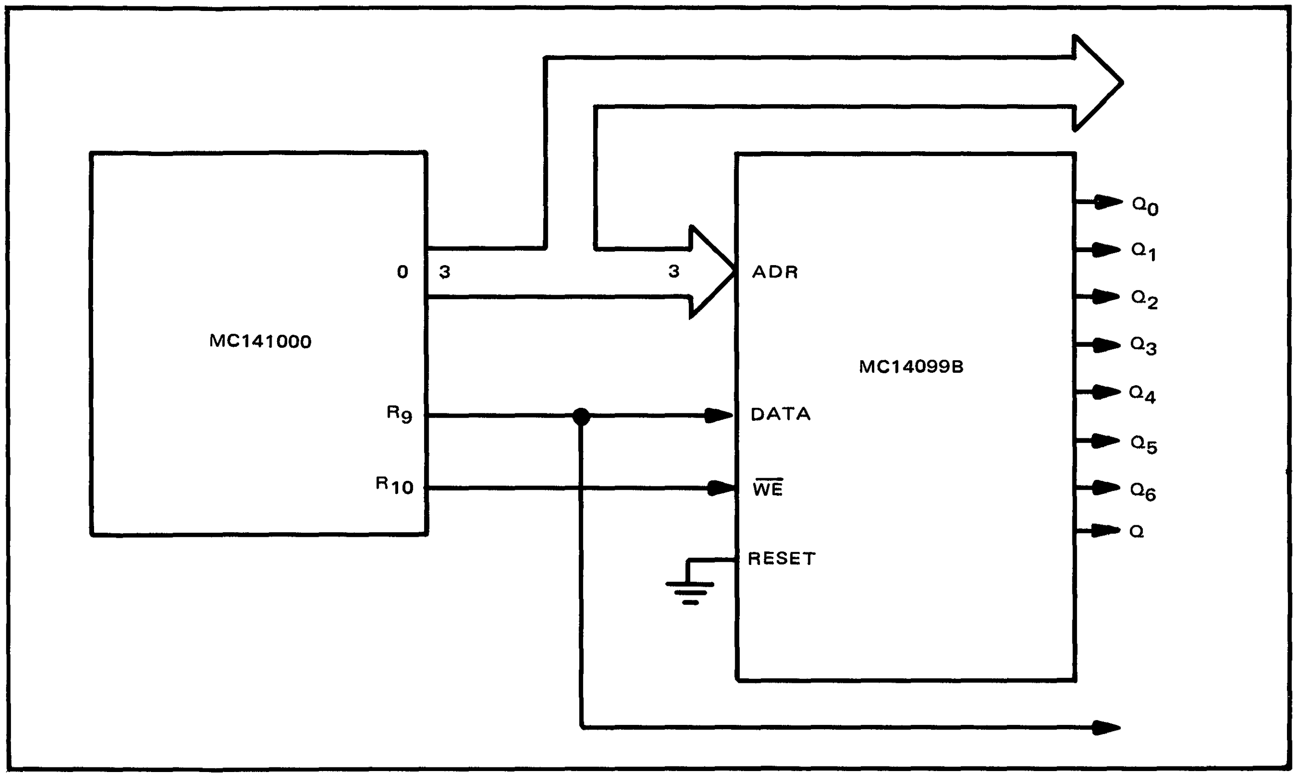
```
*****************************************************************************
SUBROUTINE TO SCAN AN LCD DISPLAY AND A 24-KEY KEYBOARD. SIX
DIGITS ARE ASSUMED HERE, BUT UP TO FOURTEEN CAN BE
ACCOMMODATED MERELY BY CHANGING THE CONSTANT ON THE TCY
INSTRUCTION.

THE DISPLAY DATA IS ASSUMED TO BE IN RAM FILE 0 WITH Y=5 BEING
THE LSD.

IF Y=15 ON RETURN, NO KEY WAS DEPRESSED. IF A KEY WAS
DEPRESSED, THE Y VALUE WILL BE IN LOCATION 14 AND THE K-INPUTS
WILL BE IN LOCATION 15 UPON RETURN.
*****************************************************************************

SCAN    LDX     0       ADDRESS THE DISPLAY DATA.
        TCY     5
NEXT    TMA             TRANSFER THE DATA TO THE O-OUTPUTS
        TDO
        SETR            LATCH THE DATA INTO THE LCD DRIVER
        KNEZ            CHECK IF KEY DOWN, IF YES, GO TO KEY DOWN
        BR      KEYDN     ROUTINE.
        RSTR
NOKEY   DYN             SELECT NEXT DIGIT, OR IF DONE, RETURN
        BR      NEXT
        RETN
KEYDN   TYA             TRANSFER Y VALUE TO ACCUMULATOR
        TCY     14      SET Y = 14
        TAMIY           STORE CURRENT Y VALUE, INCREMENT Y
        TKA             TRANSFER K INPUTS TO ACCUMULATOR
        TAM             K INPUTS TO MEMORY AT Y = 15
        TYA             15 TO ACCUMULATOR FOR DELAY
WAIT    DYN             DELAY 34*16=544 INSTRUCTION CYCLES TO
        BR      WAIT      BE CERTAIN THE KEY IS DEPRESSED AND
        DAN               NOT DUE TO BOUNCE
        BR      WAIT
        TKA             K INPUTS TO ACCUMULATOR
        SAMAN           SUBTRACT NEW K INPUTS FROM PREVIOUS
        TCY     14      ADDRESS STORED Y VALUE
        TMY             STORED Y VALUE TO Y REGISTER
        RSTR            RESET CURRENT R-LINE
        DAN             DECREMENT K INPUTS DIFFERENCE
        BR      NOKEY   BRANCH IF NOT ZERO (K INPUTS WERE NOT SAME)
        RETN            VALID KEY DEPRESSION, RETURN
```

Expanding the number of R-Outputs, Hardware and Software



Note that R9 & O's are still usable for other functions except while executing SETQ or RSTQ routines. This scheme is easily expandable to 32 Q's while dedicating only 4 R's, (for the WE lines) for a net gain of 28 outputs. The output PLA would of course need to be coded appropriately.

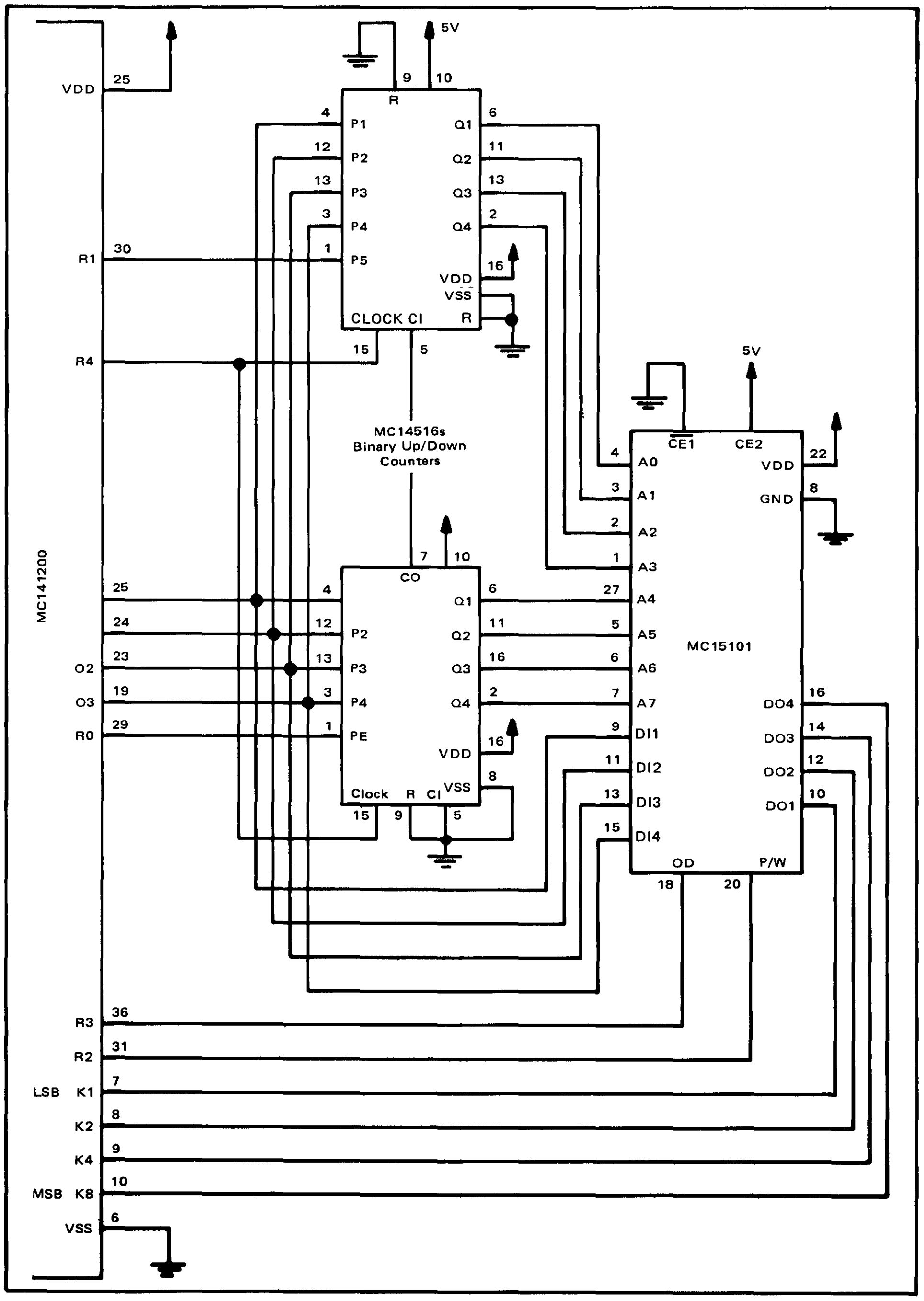
```
*****************************************************************************************************************
ROUTINE TO EXPAND R-LINES USING MC14099B 8-BIT ADDRESSABLE
LATCH. (NEW OUTPUTS WILL BE REFERRED TO AS Q-LINES IN THE
ROUTINE AND THE ACCOMPANYING DIAGRAM)

CALLING SEQUENCE: (ASSUME THE NUMBER OF THE DESIRED Q-LINE IS
IN THE ACC. THE USE OF R9 & R10 IS ARBITRARY. IT IS ASSUMED THAT
R10 HAS BEEN INITIALIZED TO A ONE AND SL TO A ZERO.)

        CALL   SETQ
(OR)
        CALL   RSTQ
********************************************************************************

SETQ    TDO                  ADDRESSES LATCH
        TCY    9             SET DATA LINE = 1
        SETR
        BR     CLOKIT
RSTQ    TDO                  ADDRESSES LATCH
        TCY    9             SET DATA LINE = 0
        RSTR
CLOKIT  TCY    10            STROBE THE DATA INTO THE LATCH
        RSTR
        SETR
        RETN
*
*
        END
```

External RAM Storage Hardware and Software



THE FOLLOWING THREE SUBROUTINES ARE USED FOR INTERFACING EXTERNAL DATA STORAGE RAM TO THE MC141000/1200.

SUBROUTINE TO SET EXTERNAL RAM ADDRESS

```
*
ADR   TCY   0    FETCH LEAST SIGNIFICANT BYTE OF ADDRESS
      TMA           FROM THE INTERNAL RAM AND OUTPUT IT
      TDO           ON THE O-LINES
      SETR       LATCH THIS ADDRESS INTO THE EXTERNAL
      RSTR          MEMORY ADDRESS REGISTER
      TCY   1    FETCH MOST SIGNIFICANT BYTE OF ADDRESS
      TMA           FROM THE INTERNAL RAM AND OUTPUT IT
      TDO           ON THE O-LINES
      SETR       LATCH THIS ADDRESS INTO THE EXTERNAL
      RSTR          MEMORY ADDRESS REGISTER
      RETN
```

```
* SUBROUTINE TO WRITE A SINGLE RAM LOCATION
*
WRITE  TCY   2    FETCH THE DATA TO BE WRITTEN FROM THE
       TMA           INTERNAL RAM AND OUTPUT IT ON THE
       TDO           O-LINES
       RSTR       WRITE THE DATA INTO THE EXTERNAL RAM
       SETR
*
* SUBROUTINE TO READ A SINGLE RAM LOCATION
*
READ   TCY   3    ENABLE THE OUTPUT DATA BUS OF THE
       RSTR          EXTERNAL RAM
       TKA        TRANSFER THE READ DATA INTO THE
       TAM           INTERNAL RAM
       SETR       DISABLE THE OUTPUT DATA BUS OF THE
       RETN          EXTERNAL RAM
*
```

MOTOROLA *Semiconductor Products Inc.*

P.O. BOX 20912 • PHOENIX, ARIZONA 85036 • A SUBSIDIARY OF MOTOROLA INC.

13586 PRINTED IN USA 4-78 IMPERIAL LITHO B20930 12M