I R O N I C S

IV1625 FOUR-PORT
SERIAL I/O BOARD

U s e r G u i d e

IRONICS
Incorporated

IV1625 FOUR-PORT
SERIAL I/O BOARD

Description: IV1625-1.002 manual
Ironics part number: 700008
Release date: 03-86

This manual is intended for use with IV1625 boards of revision level 1.0.

This manual has been carefully checked for accuracy. We can, however, assume no responsibility for errors, nor can we assume any liability which arises from the use or application of this board.

As part of our continuing effort to improve the quality of the Ironics VMEbus productivity series, we solicit comments, criticism and suggestions of our customers. We would greatly appreciate your taking the time to provide us with any feedback. Your comments, positive or negative, about the manual, hardware or software are especially welcome. Send your remarks to:

Ironics Incorporated
Quality Assurance
798 Cascadilla Street
Ithaca, New York 14850

CONTENTS

0. NOTES TO USERS OF IRONICS PRODUCTS..... 0-1
0.1 How to use this manual..... 0-3

1. GENERAL INFORMATION..... 1-1
1.1 History and System Integration..... 1-3
1.2 Features..... 1-3
1.3 Specifications..... 1-4
1.4 Technical References..... 1-5

2. INSPECTION, INSTALLATION, AND CHECKOUT..... 2-1
2.1 Unpacking Instructions..... 2-3
2.2 Inspection..... 2-3
2.3 Installation..... 2-3
2.4 Functional checkout..... 2-6
2.4.1 Hardware..... 2-6
2.4.2 Software..... 2-6

3. THEORY OF OPERATION..... 3-1
3.1 Board Control Logic..... 3-3
3.2 Addressing..... 3-3
3.3 Timing..... 3-4
3.4 Interrupter..... 3-4
3.5 Serial I/O..... 3-5

4. CONFIGURATION GUIDE..... 4-1
4.1 Base Address Selection..... 4-3
4.2 SCC Registers..... 4-3
4.3 Interrupter..... 4-4
4.4 SIO Module Special Signals..... 4-5

5. IV1625 IN THE UNIX ENVIRONMENT..... 5-1

6. INTEGRATION WITH THE IV1600/s..... 6-1

Appendix A PARTS LIST..... A-1

Appendix B BOARD LAYOUT..... B-1

Appendix C PLA TABLES..... C-1

Appendix D SCC INITIALIZATION..... D-1

Appendix E SHUNT DESCRIPTION..... E-1

Appendix F P2 PINOUT..... F-1

Appendix G IV1625 SCHEMATICS..... G-1

Appendix H IV1625 SIO MODULES..... H-1

LIST OF FIGURES

Figure 2-1. P2 Cable Installation..... 2-4

Figure 4-1. CL/RS-422 Signal Selection..... 4-5

LIST OF TABLES

TABLE 1-1. IV1625 Technical References..... 1-5

TABLE 2-1. P2 Cable Connections..... 2-4

TABLE 3-1. IV1625 PLA Descriptions..... 3-3

TABLE 3-2. SIO Module TTL Signals..... 3-5

TABLE 4-1. Addressing..... 4-3

TABLE 4-2. SCC Registers..... 4-4

TABLE 4-3. Interrupt Level Selection..... 4-4

TABLE 4-4. 20 mA/RD-B Selection..... 4-5

TABLE C-1. IV-1625 4-SIO Interupter..... C-3

TABLE C-2. IV1625 4-SIO Board select..... C-4

TABLE H-1. SIO Modules..... H-3

0. NOTES TO USERS OF IRONICS PRODUCTS

0.1 How to use this manual

This manual is intended to provide both prospective and current users of IRONICS products with the information about the IV1625 Four Port SIO Board and other multiprocessing family boards.

The manual is organized as follows:

CHAPTER 1	provides general information and systems integration notes;
CHAPTER 2	provides a procedure for installing and on site functional verification of most board features;
CHAPTER 3	provides a basic theory of board operation;
CHAPTER 4	provides detailed configuration information;
CHAPTER 5	provides UNIX application information;
CHAPTER 6	provides special integration instructions for applications involving the IRONICS IV1600 System Foundation Module.

Additional product information including chip information, selected PLA tables, and schematics is provided in the appendices.

Before attempting to integrate a board into your target system, we recommend the following steps:

[1] read the manual. familiarity with the THEORY OF OPERATION (chapter 3) will speed the process of integration;
[2] read and follow the UNPACKING, INSPECTION, and FUNCTIONAL CHECKOUT procedures outlined in chapter 2 before attempting a custom configuration;
[3] follow the configuration instructions in chapter 4 for custom configurations.

1. GENERAL INFORMATION

1.1 History and System Integration

The IV1625 4-SIO board is designed to provide VMEbus access to serial communication channels. The IV1625 plugs into the VMEbus backplane (P1 and P2) as a DTB slave module and provides 4 serial I/O channels. The board provides VMEbus A16, D8 interface and will generate interrupt requests on any VMEbus interrupt request level (IRQ1-7).

1.2 Features

The IV1625 4-SIO board provides the following features:

- Four Full Duplex Multiprotocol Serial I/O Ports;
- VMEbus Interrupter;
- Complete VMEbus DTB slave A16, D8;
- Two Z8530 Dual Channel Serial Communication Controller (SCC);
- Independent Software Programmable Baud Rates for Rx and Tx;
- Data Rates to 1 MBPS;
- Mulitdrop Monitor Modes;
- Full Modem Control;
- ASYNC, BiSYNC, SDLC Protocols;

1.3 Specifications

VMEbus Interface

DTB slave: A16, D8
Interrupter; levels 1-7, static

Addressing

PLA programmable base address, standard = 7Bxx.
Offset address is jumper selectable:

xx80	xxC0
xxA0	xxE0

PLA base addresses available are:

7Bxx	FBxx
7Fxx	FFxx

Baud Rates

Software programmable for all rates up to 1 Megabaud
Digital phase locked loop clock recovery for synchronous protocols below 256 kbaud.

Protocols

ASYNC BiSYNC SDLC

Data Encoding

NRZ NRZI BIPHASE MANCHESTER

Software Support

Unix system V driver available for full duplex asynchronous communications.

Electrical Interface

Programmable by changing interface modules. RS-232, RS-422 and Current Loop modules are available.

Physical

Double Wide Eurocard format.
Temperature - 0 to 55 deg.C operating
Humidity - 0 to 85% noncondensating

Power

+5 volts at 1.5 Amps max
+12 volts at 0.2 Amps max
-12 volts at 0.2 Amps max

1.4 Technical References

The following references will be useful to users of the IV1625. Each are available from the vendor listed.

TABLE 1-1. IV1625 Technical References

MANUAL	VENDOR
VMEbus Specification (Rev B)	Motorola
Z8530 Data Sheet	Zilog
Z8530 SCC Technical Manual	Zilog *
Z8530 SCC Application Note	Zilog
* provided by Ironics	

The VMEbus specification is a valuable reference for anyone working with the bus. It describes the electrical, mechanical and timing requirements to operate the bus. This is the specification to which the Ironics IV1625 was designed.

The data sheet addresses mostly the electrical operation of the SCC. It provides basic pinout information, internal register information and an introduction to operation.

The Z8530 technical manual provides in depth information on programming and operation of the SCC. All operating modes are explained. This reference must be read by any user who will be programming the SCC. A copy is included with the IV1625.

The Application note provides a worksheet and an example of how to initialize each communication channel.

2. INSPECTION, INSTALLATION, AND CHECKOUT

2.1 Unpacking Instructions

All IRONICS products are manufactured in a static-free environment to insure minimal degradation in to component performance due to electrical discharge. All boards are shipped in lead-shielded wrapping for protection during shipping. The following precautions should be observed before unpacking:

[1] All board handlers should be properly grounded and working in static-free work areas.
[2] Boards should be handled by board edges, avoiding contact with all connector surfaces.
[3] Avoid touching all CMOS components.

2.2 Inspection

After removing the board from its protective wrapping, visually inspect the board. Any loose debris (packing foam, etc.) should be be removed from the board surface. Inspect the following:

[1] Check all chips (EPROMs and PLAs) for loose seating. Apply even pressure on top of chip to reseat, if necessary.
[2] Check socketed chips for bent pin legs or bad connections.
[3] Check bottom of the board for broken or loose jumper wires (if present).
[4] Check the board surface for warping.

Report any serious board irregularities IMMEDIATELY to:

Ironics Incorporated
Quality Assurance
798 Cascadilla Street
Ithaca, New York 14850

(607) 277-4060

2.3 Installation

The IV1625 is shipped with four SIO modules attached to the board. The user should verify that all are seated correctly and completely. IV1625 shunts are placed according to the table in Appendix E. If possible, the user should verify that the board operates before moving any shunts. Cables are also supplied according to the SIO modules ordered. Cables should be installed on the P2 backplane, rows A and C. Appendix F describes the P2 pinout. Cables are constructed with the brown wire

indicating pin 1 (as of 6/6/85). Plans include changing to grey cable with a red stripe indicating pin 1. The following table describes the cable connections.

TABLE 2-1. P2 Cable Connections

Channel	P2 Pins	Cable Pin 1
1	A1-A16	A1
2	A17-A32	A17
3	C32-C17	C32
4	C16-C1	C16

2.4 Functional checkout

2.4.1 Hardware

The board may be checked out to insure that it operates over the VMEbus by reading its registers with the CPU board monitor. The board contains two registers per channel (8 register per board) residing on odd bytes. The following code will work with the Ironics Imon68 monitor assuming standard board address PLA's and factory shunt configuration.

```
Imon68  v1.2U> MM FF7B81;O
FF7B81  00 ? <cr>
FF7B83  00 ? <cr>
FF7B85  00 ? <cr>
FF7B87  00 ? <cr>
FF7B89  00 ? <cr>
FF7B8B  00 ? .<cr>
Imon68  v1.2U>
```

The interface to the Z8530's may be verified by writting into a r/w register and verifying the data. The example below writes the pattern AA into r/w register 12 and verifies the data.

```
Imon68  v1.2U> MM FF7B81;N
FF7B81  ?C.<cr>            { Select register 12 }
Imon68  v1.2U> MM FF7B81;N
FF7B81  ?AA.<cr>           { Load data AA }
Imon68  v1.2U> MM FF7B81;N
FF7B81  ?C.<cr>            { Select register 12 }
Imon68  v1.2U> MM FF7B81
FF7B81  AA ?.<cr>          { Read data AA }
```

2.4.1.1 SIO Modules

The IV1625 contains 4 Serial IO (SIO) modules which convert the TTL signals from the Z8530 SCC chips to proper electrical levels specified by the electrical protocol. The available SIO modules come in RS-232, RS-422 and Current Loop. The four modules ordered with the board should be properly installed on the IV1625. Be sure that the modules are seated securely.

2.4.2 Software

Software checkout should be done by coding the initialization routines found in Appendix D. The initialization routines puts the board in a state where all four ports will operate in full duplex mode, transferring data asynchronously at 9600 baud. This

initialization does not generate interrupts so the status port must be polled. The user may modify the code to change operating parameters.

To verify operation, channels 1 and 2 should be connected and channels 3 and 4 should be connected back to back. After initialization data should be transferred between the connected channels in both directions.

Following that, the user may wish to enable interrupts, at that point the SCC must be loaded with interrupt vectors, enable interrupts on the desired condition and set the master interrupt enable bit (MIE). Interrupts should ensue.

3. THEORY OF OPERATION

3.1 Board Control Logic

Much of the on-board control logic is contained within high-density programmable logic arrays (PLA's). The following table describes the PLA's on the IV1625 and their function. Complete PLA tables are provided in Appendix C.

TABLE 3-1. IV1625 PLA Descriptions

PLA #	Device Type	Designator	Description
229.0	82S153	U9	Interrupt Control
22A.0	82S153	U10	Address-R/W Decode

Logic to control timing is provides with two shift registers, (U6 & U7) and one binary counter (U15). The board also contains a small number of logic gates and flip flops used mostly for synchronization.

3.2 Addressing

The IV1625 responds to short address (A16) requests on odd bytes (DS0* asserted) and puts data on data lines D0-D8. The board decodes and responds to address modifier codes 29H and 2DH, providing short address data access for either supervisor or non-privileged accesses.

The address decode PLA (U10), monitors VMEbus signals A06-A15, address modifiers AM0, AM1, AM3, AM4 and AM5, WRITE*, DS0*, RESET* and IACK*.

Address lines A05 and A06 may by modified (inverted) by shunts to allow four boards to be addressed with the same PLA program.

When the programmed address is perceived during a DTB cycle, U10 will assert either RD or WR and the correct CE signal for the SCC being addressed. U10 monitors BLOCK* (see timing) and will hold off selecting the SCC's until it goes high.

When RESET* is asserted, U10 asserts both RD and WR providing the SCC reset condition.

Board addressing is disabled during interrupt acknowledge (IACK*) cycles.

Address line A02 drives the SCC pin A/B* to select the channel in the SCC. Address line A01 drives the SCC pin

D/C* to select the command or data register. Address line A03 is not used during address selection so A03 may be 0 or 1.

When U10 asserts a chip enable (CE) signal to the SCC the data buffer (U1) is enabled. The direction is based on the level of the VMEbus WRITE* signal. The IV1625 contains a board select LED which lights when the data buffer is enabled.

3.3 Timing

The board requires the VMEbus SYSCLK signal to generate DTACK, control access to the SCC's and to supply the required clock, PCLK, for the Z8530 SCC's.

PCLK is derived by dividing the 16Mz SYSCLK by four via the binary counter (U15). PCLK is a 4 MHz signal.

DTACK is generated via a shift register (U6). It is asserted 8 system clock pulses (500 nsec) after the data buffer (U1) is enabled connecting the SCC to the bus.

The Z8530 requires 6 PCLK cycles plus 200 nsec (1.70 microseconds) for precharge time between bus transactions involving the SCC. This is implemented with the BLOCK* signal generated from a shift register (U7). When BLOCK* is low, all accesses to the SCC's are held off. When the board is addressed, BLOCK* stays high until DTACK is generated, then BLOCK* stays low for 1.75 microsecond blocking off further board access. After that period BLOCK* goes high and any reads or writes may be serviced.

During read and write cycles, the RD and WR SCC signal assertion is held off for 125 nanoseconds after CE is asserted and the data buffer is enabled.

During a write cycle, the WR SCC signal is de-asserted concurrently with DTACK being asserted.

During a read cycle, the RD and CE SCC signals remain valid after DTACK is asserted and until the VMEbus signal DS0* is removed by the VMEbus master.

3.4 Interrupter

Each SCC may generate VMEbus interrupt requests via their INT pins. Interrupts are combined and connected to IRQ1-7 via shunts J5-11. The SCC's have an INTACK pin on which interrupts are acknowledged. When INTACK and RD are asserted, the SCC will put a pre-programmed interrupt

vector on data lines D0-D8.

The interrupt control PLA (U9) monitors interrupt requests from the SCC's INT pins. When an interrupt handler on the bus asserts interrupt acknowledge (IACK*), U9 verifies that an SCC interrupt is pending and the VMEbus acknowledge is at the correct level, by comparing A1, A2 & A3 to J19, J16 & J13. Following that, U9 will assert INTACK and RD for the interrupting SCC and enable the data buffer so the SCC can put the vector on the bus.

3.5 Serial I/O

The I/O pins for each channel of the SCC are wired to sockets into which Ironics Serial I/O modules are inserted. The modules drive or are driven by the following SCC signals as indicated by the I/O (input or output) column. This end of the SIO modules is referred to as the TTL level side.

TABLE 3-2. SIO Module TTL Signals

SCC Signal	I/O	SIO Pin	Description
-	-	2	+5V
Rx	I	3	Received Data
-	-	4	-12V
Tx	O	5	Transmitted Data
-	-	6	+12V
CTS	I	7	Clear To Send
RTS	O	9	Request To Send
DTR	O	11	Data Terminal Ready
DCD	I	13	Data Carrier Detected
TRxC	I/O	17	Transmit/Receive Clock
RTxC	I	19	Receive/Transmit Clock

The SIO Modules convert the TTL level I/O lines from the SCC to the proper level signals for the electrical interface specified by the module. Currently RS-232, RS-422 and Current Loop are supported. At the present time, the TRxC and RTxC signals are not handled by the SIO modules. These signals, if required, may be jumpered from the SIO input to outputs. Ordering information for SIO modules is presented in Appendix H.

The second socket the SIO module is inserted into, drives signals on rows A and C of the VMEbus P2 connector. Cables specified for the particular electrical interface are connected to P2 and may be mounted on the system

enclosure. Different modules may be mixed on an IV1625. Appendix H contains ordering information.

Space is provided (P12, P13, P14, P15) to allow insertion of a row of mass term pins (.1 x .1) to the electrical interface end of the SIO modules. A mass term cable to a DB-25 connector may be plugged in and brought out the front of the board. This eliminates the need for P2 connections. The mass term connectors may be assembled on the board at the factory, please specify the **on board mass term option** when ordering.

4. CONFIGURATION GUIDE

4.1 Base Address Selection

The standard base address for the IV1625, as decoded by the address decode PLA (U10) and shunts as shipped is set to 7B80. The board contains shunts to allow up to four boards to be addressed with the standard PLA. The following table describes the shunt selection assuming a standard PLA.

TABLE 4-1. Addressing

Shunt				Base Address
J15	J14	J18	J17	
in	out	in	out	7B80
out	in	in	out	7BA0
in	out	out	in	7BC0
out	in	out	in	7BE0

4.2 SCC Registers

Each SCC channel has two read/write registers that are used to access 16 internal write registers and 9 internal read registers. The first r/w register is the command register. When read, it provides the data in internal register RR0, the status register. When written the data goes into register WR0, the command register. Writing to the command register (WR0) attaches another register to the command register for reading and writing. For example writing a 3 to WR0 would select internal register 3 (RR3 & WR3), so if the next cycle is a write the data written to the command register (WR0) would go into WR3. Likewise if the next cycle is a read, the data read from the command register (RR0) would be the contents of RR3.

Reading the command register without first writing a register yields the contents of the status register (RR0).

The other register, called the data register, provides direct read/write access to the internal receive (RR8) and transmit (WR8) registers.

The following table describes the registers and their addressing as shipped.

TABLE 4-2. SCC Registers

Channel	Register	Address
1	Command	7B85
1	Data	7B87
2	Command	7B81
2	Data	7B83
3	Command	7B95
3	Data	7B97
4	Command	7B91
4	Data	7B93

4.3 Interrupter

The VMEbus level that interrupt requests will be directed to may be selected by the user. This level is reflected with shunts J5-J11, J13, J16 and J19. Only one shunt between J5 and J11 can be inserted to assure proper interrupt operation. If J5-J11 are all vacant, the board will never generate VMEbus interrupt requests and any interrupt request pending on either SCC will remain pending. The following table describes the jumpers for each interrupt request level.

TABLE 4-3. Interrupt Level Selection

Level	Insert Only	J13 (A03)	J16 (A02)	J19 (A01)
IRQ1	J11	in	in	out
IRQ2	J10	in	out	in
IRQ3	J9	in	out	out
IRQ4	J8	out	in	in
IRQ5	J7	out	in	out
IRQ6	J6	out	out	in
IRQ7	J5	out	out	out

4.4 SIO Module Special Signals

In order to fit the signals for all electrical interfaces, two signals, 20 milliamp Current Source for the Current Loop module and RD-B for the RS-422 module, are multiplexed on one P2 pin. Shunts J1-J4 configure the signal selection for channels 1 to 4. The shunt position does not matter if RS-232 SIO modules are being used. The shunts are configured in a 1x3 fashion, and with bezel side of the board facing you, pin 1 is to the left. There are two positions, 1 and 2 shown in the following figure.

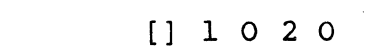


Figure 4-1. CL/RS-422 Signal Selection

The following table describes the shunts that must be inserted to match the SIO module used.

TABLE 4-4. 20 mA/RD-B Selection

Channel	Module	Signal	Shunt
1	Current Loop	20 mA	J1 position 1
1	RS-422	RD-B	J1 position 2
2	Current Loop	20 mA	J2 position 1
2	RS-422	RD-B	J2 position 2
3	Current Loop	20 mA	J3 position 1
3	RS-422	RD-B	J3 position 2
4	Current Loop	20 mA	J4 position 1
4	RS-422	RD-B	J4 position 2

5. IV1625 IN THE UNIX ENVIRONMENT

The IV1625 may operate in the UNIX environment. Interrupt handling under UNIX requires modification to the UNIX kernel. The Ironics Unix development system provides a means for modifying the kernel in such a manner to allow the integration of new device drivers. The driver consists of seven routines, initialization, open, close, read, write, control and interrupt.

The Initialization routine resets the on board chips (SCC's) and puts the board into a standard state. Each channel is activated for asynchronous operation at 9600 baud, 8 bits/character, 1 stop bit. The internal baud rate generator is selected (x 16 clock) and the output is sent to the TRxC pin (for debug purposes). Interrupts for Rx, Tx, Ext/Sta are not enabled but the MIE (Master Interrupt Enable) is enabled. The interrupt vector is set for each channel and the vector is modified to reflect status.

During Open, DTR and RTS are turned on, Autoenables is turned on and Rx and Tx interrupts are enabled.

The Close routine turns off RTS and DTR and disables Rx and Tx interrupts.

Read and Write call the UNIX terminal line switch which calls the proc routine to actually do the I/O.

The Control routine calls the param routine to set the termio(7) parameters.

There are four Interrupt routines. The parameter passed to then by the interrupt dispatch identifies the device (channel) on which the interrupt occurred. Receive and Transmit interrupt routines are in place. Routines for EXT/STA and SPEC conditions do no processing but simply reset the interrupt. The SCC's are not programmed to generate these interrupts and under normal circumstances they should not occur. As shipped, the IV1625 is configured to interrupt on request level IRQ2, this is a requirement for operation under UNIX.

The driver has been integrated into the kernel and occupies the slot for major device number 7. The minor devices are 0-3 for the first board and 4-7 for the second board and so on.

Ironics will supply a kernel with the drivers for 1 IV1625 integrated into it. The kernel should be ordered with the system to allow factory checkout. Field upgrades are not supported. Other kernels or configuration may be proposed to the System Support Group.

Reconfiguration source code includes the standard IV1600 drivers and configuration files in addition to the IV1625 drivers. The IV1625 code is contained in two files, a header file, 1625.h and the 'C' file, 1625.c.

It is important to remember that having an IV1625 operating in the UNIX environment does not make the system capable of supporting 8 users. The system is optimized for 4 users and serious degradation will occur with more that the original 4 users.

6. INTEGRATION WITH THE IV1600/s

In order to operate the IV1625 in the VMEbus card cage supported by the IV1600 several modification may have to be made.

If the IV1625 is to generate interrupts the IACKIN/IACKOUT daisy chain must be in place. This is done by inserting a jumper on the backplane pins for each slot between the IV1600 and the IV1625 which is unoccupied or occupied by a board which does not contain an interrupter (i.e. a RAM board), not including the slot which contains the IV1625.

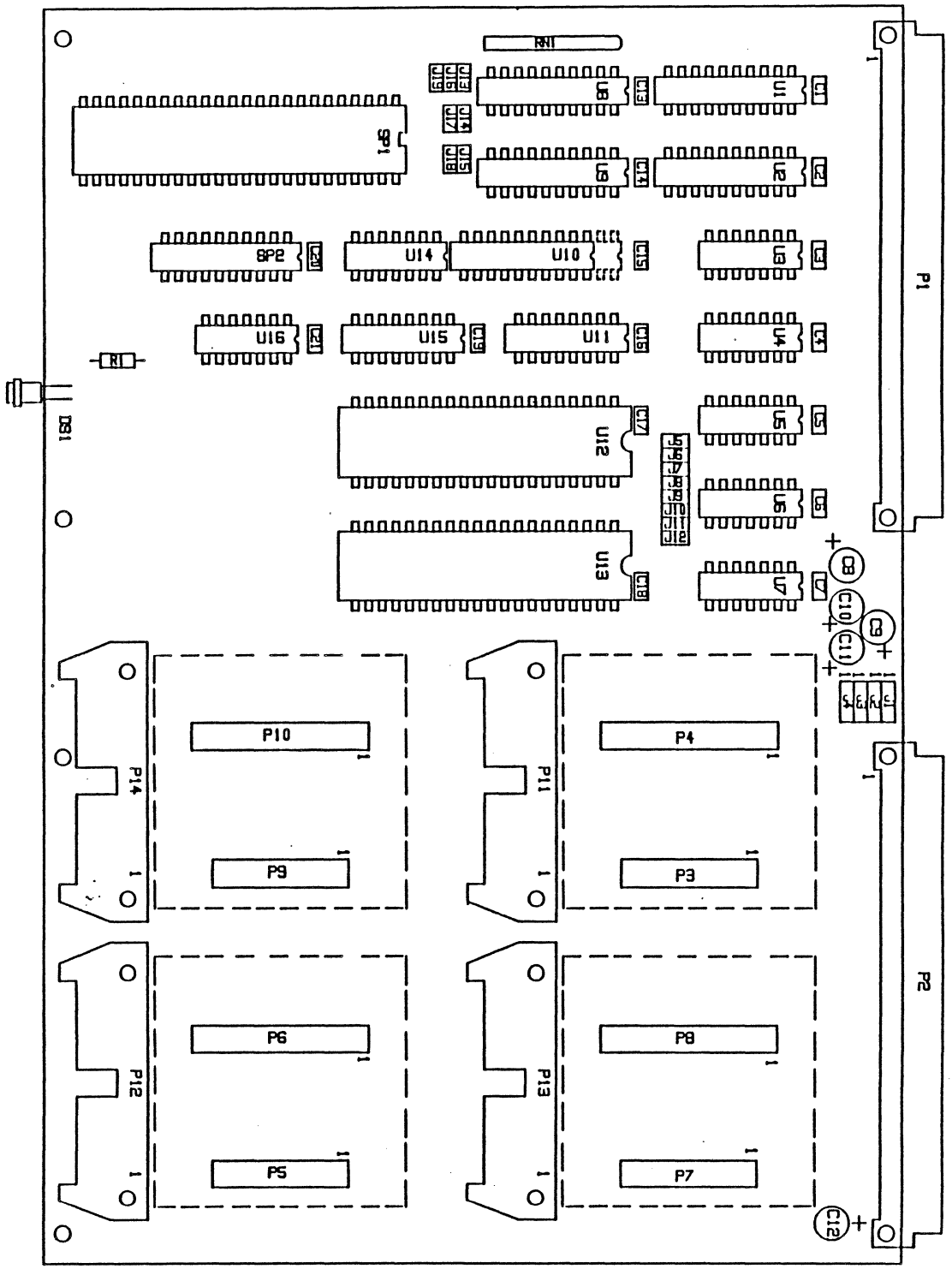
The IV1600 must be enables via software (IMASK*) and hardware (shunts) to accept VMEbus interrupts. Note also that U26 (IV1600, rev 1.3) must be PLA program revision 233.1 or greater.

As shipped the IV1625 is configured to interrupt at request level IRQ2. This is a requirement for operation under UNIX.

Appendix A PARTS LIST

Mfr Pn	Ref des	Description
74F245	U1	octal transceiver
74F244	U2	octal buffer
74F20	U3	dual 4 input NAND
74F10	U4	tri 3 input NAND
74F38	U5	quad 2 input NAND
74LS164	U7,U6	8 bit shift register
74F240	U8	octal inverter buffer
82S153	U9	Interrupter 229.0
82S153	U10	Bus Interface 22A.0
74LS375	U11	quad bistable latch
Z8530	U12,U13	Dual SCC
74F74	U14	dual flip flop
74LS161	U15	4 bit binary counter
74F00	U16	quad 2 input NAND
	P3,P5,P7,P9	2 strips of 10 socket pins on .1"
	P4,P6,P8,P10	2 strips of 13 socket pins on .1"
	P11,P12,P13,P14	26 pin dual row header
	P1,P2	DIN VME connectors
	J1-J7	7 position dual row header
	J8,J9	2 2 position, dual row headers
	J12-J14	3 position, dual row header
	J15-J22	3 3 pos., single row headers
	J23	strip of 2 header posts
	C1-C5	10 ufd./16V radial electrolytic
	C6-C21	.1 ufd. dipguard
	CR1	MV5454A Green LED
	RN1	2.2K, 10 pin SIP
	R1	470 ohm .25W 5%
	sU12,sU13	40 pin socket
	sU9,sU10	20 pin socket
C01-1206-1.0		Printed Circuit Board Bezel+Hardware

Appendix B BOARD LAYOUT


RN1
U8
U1
U9
U2
SP1
8P2
U14
U10
U3
U16
U15
U11
U4
RN
DS1
U12
U5
U6
U13
U7
C8
C10
C11
C3
C12
P1
P2
P10
P4
P14
P11
P9
P3
P6
P8
P12
P13
P5
P7

Appendix C PLA TABLES

TABLE C-1. IV-1625 4-SIO Interupter

```
U9
229.0
82S153
                                                    *POL HLLLLHHHHL
*P 00 *I ---LLLHH *BI L--------- *BO .A........
*P 01 *I H--LLL-- *BI L----L---- *BO .A........
*P 02 *I -H-LLL-- *BI L-----L--- *BO .A........
*P 03 *I --HLLL-- *BI L------L-- *BO .A........
*P 04 *I L--LLL-- *BI L----H---- *BO .A........
*P 05 *I -L-LLL-- *BI L-----H--- *BO .A........
*P 06 *I --LLLL-- *BI L------H-- *BO .A........
*P 07 *I LLH----- *BI -----LLH-- *BO .........A
*P 08 *I LHL----- *BI -----LHL-- *BO .........A
*P 09 *I LHH----- *BI -----LHH-- *BO .........A
*P 10 *I HLL----- *BI -----HLL-- *BO .........A
*P 11 *I HLH----- *BI -----HLH-- *BO .........A
*P 12 *I HHL----- *BI -----HHL-- *BO .........A
*P 13 *I HHH----- *BI -----HHH-- *BO .........A
*P 14 *I ---LLLHL *BI L--------L *BO ...AA.....
*P 15 *I ---LLLLH *BI L--------L *BO ..A.A.....
*P 16 *I ---LLLLL *BI L--------L *BO ..A.A.....
*P 17 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 18 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 19 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 20 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 21 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 22 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 23 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 24 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 25 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 26 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 27 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 28 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 29 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 30 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 31 *I 00-000-0 *BI -00-000000 *BO AAAAAAAAAA
*P D9 *I 00000000 *BI 0000000000
*P D8 *I -------- *BI ----------
*P D7 *I -------- *BI ----------
*P D6 *I -------- *BI ----------
*P D5 *I -------- *BI ----------
*P D4 *I 00000000 *BI 0000000000
*P D3 *I 00000000 *BI 0000000000
*P D2 *I 00000000 *BI 0000000000
*P D1 *I 00000000 *BI 0000000000
*P D0 *I -------- *BI ----------
```

TABLE C-2. IV1625 4-SIO Board select

```
Oct 16, 1984
U10
22A.0
82S153
                                                    *POL HHHLHLLHHH
*P 00 *I LLLLLLLL *BI HHH----LLH *BO ....AA....
*P 01 *I LLLLLLLH *BI HHH----LLH *BO ....A.A...
*P 02 *I LLLLLLLL *BI HHH----HLH *BO ...A.A....
*P 03 *I LLLLLLLH *BI HHH----HLH *BO ...A..A...
*P 04 *I -------- *BI ---------L *BO ...AA.....
*P 05 *I -------- *BI -----L-LLH *BO ....AA....
*P 06 *I -------- *BI ------LLLH *BO ....A.A...
*P 07 *I -------- *BI -----L-HLH *BO ...A.A....
*P 08 *I -------- *BI ------LHLH *BO ...A..A...
*P 09 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 10 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 11 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 12 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 13 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 14 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 15 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 16 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 17 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 18 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 19 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 20 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 21 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 22 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 23 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 24 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 25 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 26 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 27 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 28 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 29 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 30 *I 00000000 *BI 0000000000 *BO AAAAAAAAAA
*P 31 *I 00-000-0 *BI -0-0000000 *BO AAAAAAAAAA
*P D9 *I 00000000 *BI 0000000000
*P D8 *I 00000000 *BI 0000000000
*P D7 *I 00000000 *BI 0000000000
*P D6 *I -------- *BI ----------
*P D5 *I -------- *BI ----------
*P D4 *I -------- *BI ----------
*P D3 *I -------- *BI ----------
*P D2 *I 00000000 *BI 0000000000
*P D1 *I 00000000 *BI 0000000000
*P D0 *I 00000000 *BI 0000000000
```

Appendix D SCC INITIALIZATION

```
/* 1625.h              IV-1625 4-SIO Declarations 30 Oct 1984
 *
 * Copyright 1984  Kevin J. Lynch, Ironics Incorporated
 */


/* Each IV-1625 contains two Z8530 SCC chips. Each chip contains two
 * ports.  Each port is represented by two 8 bit registers
 */
struct device {
        char s25_cont; /* control register */
        char s25_xxx2;  /* filler */
        char s25_data;  /* data register */
        };

/* The following describes the character driver tables for the
 * IV-1625 4-SIO VMEbus card.  Each card contains 2 Z8530 SCC's.
 */

#define S25ADDR                 0xFF7B80        /* boards base address
#define S25CNT                  4               /* Number of channels
int s25_cnt = S25CNT;
unsigned char s25_modem[S25CNT];
struct tty s25_tty[S25CNT];
struct ttyptr s25_ttptr[] = {
        S25ADDR+0x05,  &s25_tty[0],             /* chip 1 channel A */
        S25ADDR+0x01,  &s25_tty[1],             /* chip 1 channel B */
        S25ADDR+0x15,  &s25_tty[2],             /* chip 2 channel A */
        S25ADDR+0x11,  &s25_tty[3],             /* chip 2 channel B */
};

#define S25_CLKRATE     4000000 /* 4 Mhz clock rate */
#define S25_VECTOR      0x5A    /* Composite interrupt vector */

/* Read register zero values */
/* Transmit/Receive buffer status and External status */
#define R0_RXRDY        0x01    /* Rx character available */
#define R0_ZERO         0x02    /* Zero count */
#define R0_TXEMP        0x04    /* Tx Buffer empty */
#define R0_DCD          0x08    /* DCD */
#define R0_SYNHUN       0x10    /* Sync - Hunt */
#define R0_CTS          0x20    /* CTS */
#define R0_TXUND        0x40    /* Tx underrun - EOM */
#define R0_BREAK        0x80    /* Break - Abort */

/* Read register one values */
/* Special receive Condition Status */
#define R1_SENT         0x01    /* All sent */
#define R1_RC2          0x02    /* Residue code 2 */
#define R1_RC1          0x04    /* Residue code 1 */
```

```
#define R1_RC0          0x08    /* Residue code 0 */
#define R1_PARER        0x10    /* Parity Error */
#define R1_RXOVER       0x20    /* Rx overrun error */
#define R1_FRAME        0x40    /* Framing error */
#define R1_CRC          0x40    /* CRC error */
#define R1_EOF          0x80    /* End of frame (sdlc) */

/* Read register two
 * Modified interrupt vector channel B, Unmodified channel A
 * bits 3       2       1       Status high/Status low = 0
 * bits 4       5       6       Status high/Status low = 1
 *
 *      0       0       0       B Tx empty
 *      0       0       1       B Ext/status change
 *      0       1       0       B Rx ready
 *      0       1       1       B Rx special condition
 *      1       0       0       A Tx empty
 *      1       0       1       A Ext/status change
 *      1       1       0       A Rx ready
 *      1       1       1       A Rx special condition
 */


/* Read register three values */
/* Interrupr Pending bits */
#define R3_BEXT         0x01    /* Channel B EXT - Stat IP */
#define R3_BTXIP        0x02    /* Channel B Tx IP */
#define R3_BRXIP        0x04    /* Channel B Rx IP */
#define R3_AEXT         0x08    /* Channel A EXT - Stat IP */
#define R3_ATXIP        0x10    /* Channel A Tx IP */
#define R3_ARXIP        0x20    /* Channel A Rx IP */

/* Read register eight */
/* Receive buffer */

/* Read register ten values */
/* Miscellaneous status */
#define R10_OLOOP       0x02    /* On loop */
#define R10_SLOOP       0x10    /* Loop sending */
#define R10_TCM         0x40    /* Two clocks missing */
#define R10_OCM         0x80    /* One clock missing */

/* Read register twelve */
/* Lower byte of baud rate generator time constant */

/* Read register thirteen */
/* Upper byte of baud rate generator constant */

/* Read register fifteen values */
/* External/Status interrupt information */
#define R15_IEZ         0x02    /* Zero count IE */
```

```
#define R15_IEDCD       0x08     /* DCD IE */
#define R15_IESYNC      0x10     /* Sync - Hunt IE */
#define R15_IECTS       0x20     /* CTS IE */
#define R15_IETX        0x40     /* Tx underrun - EOM IE */
#define R15_IEBRK       0x80     /* Break - Abort IE */

/* Write register zero values */
/* CRC initialize, initialization commands for modes, Register pointer
#define W0_AR0          0x0      /* Set to access register 0 */
#define W0_AR1          0x1      /* Set to access register 1 */
#define W0_AR2          0x2      /* Set to access register 2 */
#define W0_AR3          0x3      /* Set to access register 3 */
#define W0_AR4          0x4      /* Set to access register 4 */
#define W0_AR5          0x5      /* Set to access register 5 */
#define W0_AR6          0x6      /* Set to access register 6 */
#define W0_AR7          0x7      /* Set to access register 7 */
#define W0_AR8          0x8      /* Set to access register 8 */
#define W0_AR9          0x9      /* Set to access register 9 */
#define W0_AR10         0xa      /* Set to access register 10 */
#define W0_AR11         0xb      /* Set to access register 11 */
#define W0_AR12         0xc      /* Set to access register 12 */
#define W0_AR13         0xd      /* Set to access register 13 */
#define W0_AR14         0xe      /* Set to access register 14 */
#define W0_AR15         0xf      /* Set to access register 15 */

#define W0_REI          0x10     /* Reset Ext - Status interrupts */
#define W0_SNDAB        0x18     /* Send Abort */
#define W0_EIRX         0x20     /* Enable interrupts on next Rx */
#define W0_RTXI         0x28     /* Reset Tx int pending */
#define W0_RERR         0x30     /* Error Reset */
#define W0_RHIUS        0x38     /* Reset highest IUS */
#define W0_RRXCRC       0x40     /* Reset Rx CRC */
#define W0_RTXCRC       0x80     /* Reset Tx CRC */
#define W0_RTXUND       0xc0     /* Reset Tx underrun - EOM latch */

/* write register one values */
/* Transmit/Receive interrupt and data transfer mode definition */
#define W1_EXTIE        0x01     /* Ext interrupt enable */
#define W1_TXIE         0x02     /* Tx interrupt enable */
#define W1_PARSPE       0x04     /* Parity is special condition */
#define W1_RXID         0x00     /* Rx interrupt disabled */
#define W1_RX1IE        0x08     /* Rx interrupr on lst char or special
#define W1_RXAIE        0x10     /* RX interrupt on all char or special
#define W1_RXSIE        0x18     /* RX on special only */

/* write register two */
/* Interrupt vector (accessed through either channel) */

/* write register three values */
/* Receive parameters and control */
#define W3_RXENA        0x01     /* Rx enabled */
```

```
#define W3_SCLI         0x02     /* Sync character load inhibit
#define W3_ASM          0x04     /* Address search mode */
#define W3_RXCRCE       0x08     /* Rx CRC enabled */
#define W3_HUNT         0x10     /* Enter hunt mode */
#define W3_AUTOE        0x20     /* Auto enables */
#define W3_R5BIT        0x00     /* Rx 5 bits/char */
#define W3_R7BIT        0x40     /* Rx 7 bits/char */
#define W3_R6BIT        0x80     /* Rx 6 bits/char */
#define W3_R8BIT        0xc0     /* Rx 8 bits/char */

/* write register four values */
/* Transmit/Receive misc. parameters and modes */
#define W4_PARENA       0x01     /* Parity enable */
#define W4_PAREVE       0x02     /* Parity even */
#define W4_PARODD       0x00     /* Parity odd */
#define W4_SYNCENA      0x00     /* Sync modes enable */
#define W4_1STOP        0x04     /* 1 stop bit/char */
#define W4_15STOP       0x08     /* 1.5 stop bits/char */
#define W4_2STOP        0x0c     /* 2 stop bits/char */
#define W4_8SYNC        0x00     /* 8 bit sync character */
#define W4_16SYNC       0x10     /* 16 bit sync char */
#define W4_SDLC         0x20     /* SDLC mode */
#define W4_EXTSYN       0x30     /* External sync mode */
#define W4_X1CLK        0x00     /* X 1 clock */
#define W4_X16CLK       0x40     /* X 16 clock */
#define W4_X32CLK       0x80     /* X 32 clock */
#define W4_X64CLK       0xc0     /* X 64 clock */

/* write register five values */
/* Transmit parameters and controls */
#define W5_TXCRCE       0x01     /* Tx CRC enabled */
#define W5_RTS          0x02     /* RTS */
#define W5_CRC16        0x04     /* SDLC* - CRC-16 */
#define W5_TXENA        0x08     /* Tx enable */
#define W5_TXBRK        0x10     /* Send break */
#define W5_T5BIT        0x00     /* Tx 5 bits (or less)/char */
#define W5_T7BIT        0x20     /* Tx 7 bits/char */
#define W5_T6BIT        0x40     /* Tx 6 bits/char */
#define W5_T8BIT        0x60     /* Tx 8 bits/char */
#define W5_DTR          0x80     /* DTR */

/* write register six */
/* Sync characters or SDLC address field */

/* write register seven */
/* Sync character of SDLC flag */

/* write register eight */
/* Transmit buffer */

/* write register nine values */
```

```
/* Master interrupt control and reset, accessed through either channel
#define W9_VIS          0x01     /* VIS */
#define W9_NV           0x02     /* NV */
#define W9_DLC          0x04     /* DLC */
#define W9_MIE          0x08     /* MIE */
#define W9_STAHI        0x10     /* Status High */
#define W9_STALO        0x00     /* Status low */
#define W9_NORST        0x00     /* No reset */
#define W9_BRESET       0x40     /* Channel B reset */
#define W9_ARESET       0x80     /* Channel A reset */
#define W9_HRESET       0xc0     /* Force hardware reset */

/* write register 10 values */
/* Misc. Transmitter/Receiver control bits */
#define W10_S6BIT       0x01     /* 6 bit sync */
#define W10_S8BIT       0x00     /* 8 bit sync */
#define W10_LOOP        0x02     /* Loop mode */
#define W10_ABOUND      0x04     /* Abort on underrun */
#define W10_FLGUND      0x00     /* Flag on underrun */
#define W10_MRKIDL      0x08     /* Mark on idle */
#define W10_FLGIDL      0x00     /* Flag on idle */
#define W10_GAOR        0x10     /* Go Active On Roll */
#define W10_NRZ         0x00     /* NRZ */
#define W10_NRZI        0x02     /* NRZI */
#define W10_FM1         0x40     /* FM1 (transition = 1) */
#define W10_FM0         0x40     /* FM0 (transition = 0) */
#define W10_CRCI        0x80     /* CRC Preset I */
#define W10_CRCO        0x00     /* CRC Preset 0 */

/* write register eleven values */
/* Clock mode control */
#define W11_XTAL        0x00     /* TRxC out = XTAL */
#define W11_TXCLK       0x01     /* TRxC out = Transmit clock */
#define W11_BRG         0x02     /* TRxC out = BR generator */
#define W11_DPLL        0x03     /* TRxC out = DPLL output */
#define W11_OTRXC       0x04     /* TRxC Output */
#define W11_ITRXC       0x00     /* TRxC Input */
#define W11_TRTXC       0x00     /* Transmit clock = RTxC pin */
#define W11_TTRXC       0x08     /* Transmit clock = TRxC pin */
#define W11_TBRG        0x10     /* Transmit clock = BR generator */
#define W11_TDPLL       0x18     /* Transmit clock = DPLL output */
#define W11_RRTXC       0x00     /* Receive clock = RTxC pin */
#define W11_RTRXC       0x20     /* Receive clock = TRxC pin */
#define W11_RBRG        0x40     /* Receive clock = BR generator */
#define W11_RDPLL       0x60     /* Receive clock = DPLL output */
#define W11_RXTAL       0x80     /* RTxC XTAL */
#define W11_RNXTAL      0x00     /* RTxC no XTAL */

/* write register twelve */
/* Lower byte of baud rate generator time constant */
```

```
/* write register thirteen */
/* Upper byte of baud rate generator time constant */

/* write register fourteen values */
/* Misc. control bits */
#define W14_BRGENA      0x01      /* BR generator enable */
#define W14_BRGSRC      0x02      /* BR generator source */
#define W14_REQFUN      0x04      /* Request Function */
#define W14_DTR         0x00      /* DTR */
#define W14_AUTOEC      0x08      /* Auto Echo */
#define W14_LOCLPB      0x10      /* Local Loopback */
#define W14_ENTSM       0x20      /* Enter Search mode */
#define W14_RSTMC       0x40      /* Reset missing clock */
#define W14_DDPLL       0x60      /* Disable DPLL */
#define W14_SBRG        0x80      /* Source = BR generator */
#define W14_SRTXC       0x90      /* Source = RTxC */
#define W14_FM          0xc0      /* Set FM mode */
#define W14_NRZI        0xe0      /* set NRZI mode */

/* write register fifteen values */
/* external/Status interrupt control */
#define W15_ZCIE        0x02      /* Zero count Interrupt enable */
#define W15_DCDIE       0x80      /* DCD interrupt enable */
#define W15_SYNCIE      0x10      /* Sync hunt interrupt enable */
#define W15_CTSIE       0x20      /* CTS interrupr enable */
#define W15_TXUNDIE     0x04      /* Tx underrun - EOM interrupr enable
#define W15_BRKIE       0x08      /* Break-Abort interrupt enable */

/* The baud rate constant is computed in the following fashion:
 *
 *      speed = S25_CLKRATE;
 *      speed = (speed/(s25speeds[sspeed]<<1<<4)) - 2;
 *
 * where S25_CLKRATE is 4 Mhz,
 *       sspeed is defined as B9600,
 *       s25speeds contains actual baud rates,
 *       <<1 divides by 2
 *       <<4 divides by 16, because of 16X clock.
*/

int s25speeds[] = {
        1,      50,     75,     110,    134,    150,    200,    300,
        600,    1200,   1800,   2400,   4800,   9600,   19200,  38400
        };

/*
 * s25init called in sysinit to pre-initialize all ports
 */

s25init()
{
```

```
        register int port;

        for (port=0; port < s25_cnt; port++)
        {
                initl(port);
        }
        printf("%d IV-1625 ports initialized0, port);
}

initl(port)
int port;
{
        register long speed;
        char resets25();

/* modes and constants */
        writes25(port, 9, resets25(port));      /* reset port */
        writes25(port, 4, W4_X16CLK|W4_1STOP);
        writes25(port, 3, W3_R8BIT);
        writes25(port, 5, W5_T8BIT);
        writes25(port, 6, 0);                   /* Clear sync byte */
        writes25(port, 7, 0);                   /* Clear sync byte */
        writes25(port, 9, 0);                   /* Clear MIE */
        writes25(port, 10, 0);                  /* Clear sync, loop
        writes25(port, 11, W11_RBRG|W11_TBRG|W11_OTRXC|W11_BRG);
        speed = S25_CLKRATE;
        speed = (speed/(s25speeds[sspeed]<<1<<4)) - 2;
        writes25(port, 12, (unsigned char)speed);
        writes25(port, 13, (unsigned char)speed>>8);
        writes25(port, 14, W14_BRGSRC);
/* enables */
        writes25(port, 3, W3_RXENA|W3_R8BIT);
        writes25(port, 5, W5_TXENA|W5_T8BIT);
        writes25(port, 0, W0_RTXCRC);
        writes25(port, 14, W14_BRGSRC|W14_BRGENA);
/* interrupt enables */
        writes25(port, 15, 0);                  /* Clear ext IE bits *
        writes25(port, 0, W0_REI);
        writes25(port, 0, W0_REI);
        writes25(port, 2, S25_VECTOR);
        writes25(port, 9, W9_MIE);      /* master interrupt enable */
}

char resets25(port)
int port;
{
        if ((port & 1) == 1)
                return(W9_BRESET);
        else    return(W9_ARESET);
}
```

```
writes25(port, reg, outdata)
int port;
int reg;
char outdata;
{
        struct device *addr;

        addr = (struct device *)s25_ttptr[port].tt_addr;
        switch (reg)
        {
        case 0: addr->s25_cont = outdata&0xFF;
                break;
        case 8: addr->s25_data = outdata&0xFF;
                break;
        default:
                addr->s25_cont = reg&0x0F;
                addr->s25_cont = outdata&0xFF;
                break;
        }
}

unsigned char reads25(port, reg)
int port;
int reg;
{
        char indata;
        struct device *addr;

        addr = (struct device *)s25_ttptr[port].tt_addr;
        switch (reg)
        {
        case 0: indata = addr->s25_cont;
                break;
        case 8: indata = addr->s25_data;
                break;
        default:
                addr->s25_cont = reg&0x0F;
                indata = addr->s25_cont;
                break;
        }
        return(indata);
}
```

Appendix E SHUNT DESCRIPTION

Below is a table which describes the shunts on the IV1625 4-SIO board. Shunts which are inserted at the factory are indicated with a *.

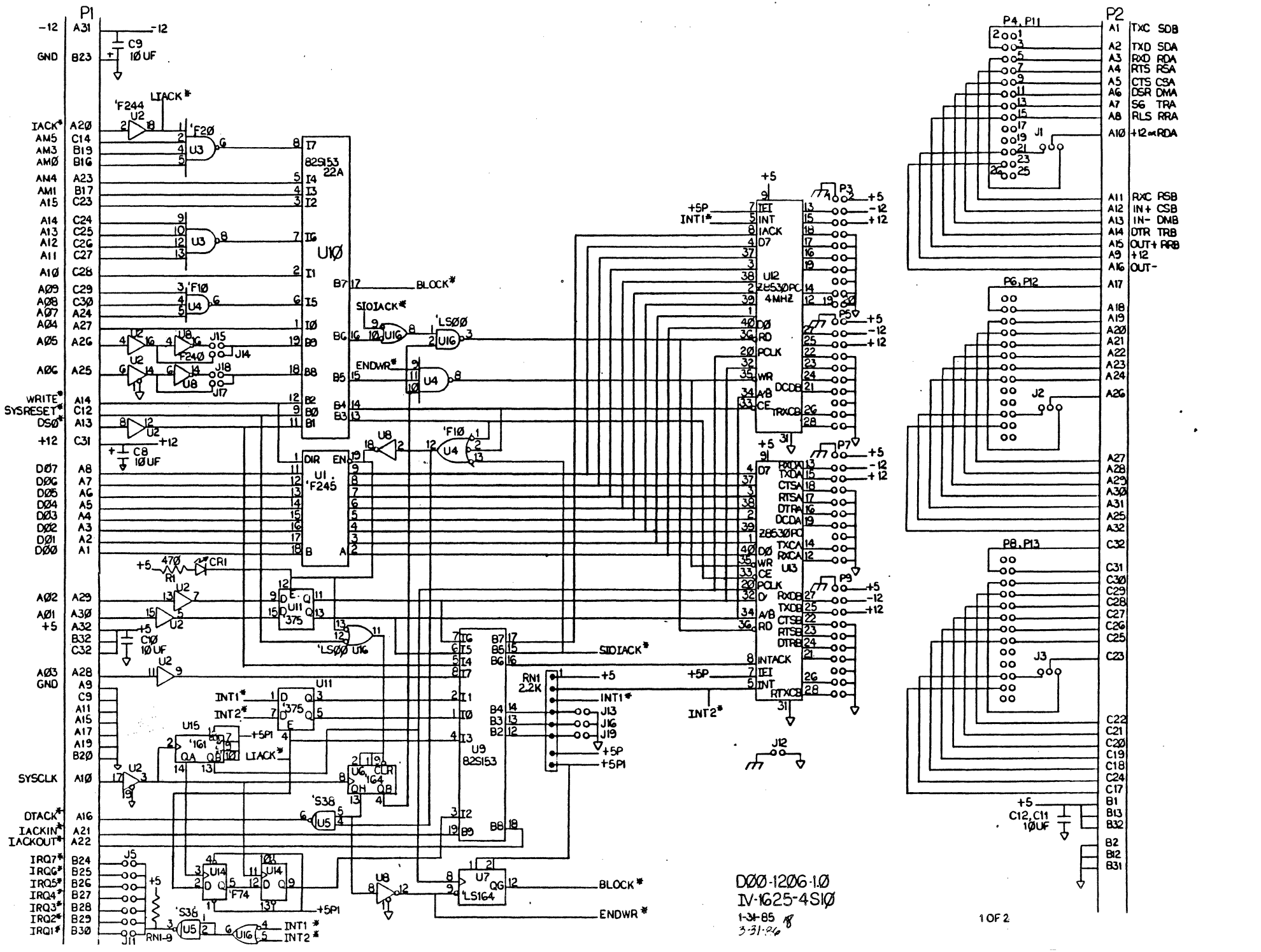
Shunt		Description
J1		Selects Channel 1 CL/RS-422
J2		Selects Channel 2 CL/RS-422
J3		Selects Channel 3 CL/RS-422
J4		Selects Channel 4 CL/RS-422
J5		Selects IRQ7
J6		Selects IRQ6
J7		Selects IRQ5
J8		Selects IRQ4
J9		Selects IRQ3
J10	*	Selects IRQ2
J11		Selects IRQ1
J12		Signal ground to Logic ground
J13	*	IRQ level comparison bit 2
J16		IRQ level comparison bit 1
J19	*	IRQ level comparison bit 0
J14		Address Selection A05 direct
J15	*	Address Selection A05 inverted
J17		Address Selection A06 direct
J18	*	Address Selection A06 inverted

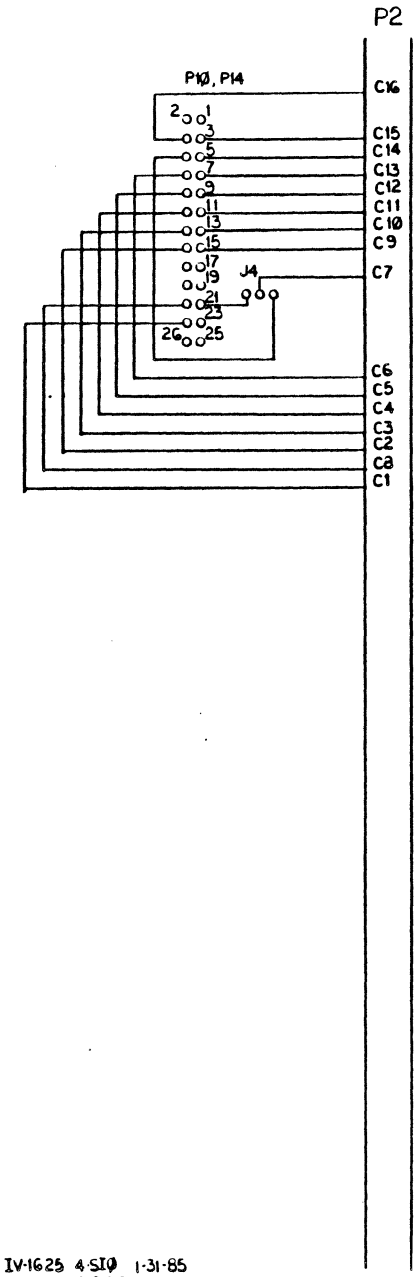
Appendix F P2 PINOUT

The following table describes the serial I/O lines present on the VMEbus P2 connector as defined as the IV-1625. Row B is defined by the VMEbus and the IV-1625 only uses +5V (B1, B13, B32) and GND (B2, B12, B31).

Pin #	Row A			Row C		
	Chan	RS-232 & C.L.	RS-422	Chan	RS-232 & C.L.	RS-422
1	1	TxC	SD-B	4	OUT-	SG
2	1	TxD	SD-A	4	OUT+	RR-B
3	1	RxD	RD-A	4	DTR	TR-B
4	1	RTS	RS-A	4	IN-	DM-B
5	1	CTS	CS-A	4	IN+	CS-B
6	1	DSR	DM-A	4	RxC	RS-B
7	1	SG	TR-A	4	CS	RD-A
8	1	RLS	RR-A	4	CS	-
9	1	CS	-	4	RLS	RR-A
10	1	CS	RD-A	4	SG	TR-A
11	1	RxC	RS-B	4	DSR	DM-A
12	1	IN+	CS-B	4	CTS	CS-A
13	1	IN-	DM-B	4	RTS	RS-A
14	1	DTR	TR-B	4	RxD	RD-A
15	1	OUT+	RR-B	4	TxD	SD-A
16	1	OUT-	SG	4	TxC	SD-B
17	2	TxC	SD-B	3	OUT-	SG
18	2	TxD	SD-A	3	OUT+	RR-B
19	2	RxD	RD-A	3	DTR	TR-B
20	2	RTS	RS-A	3	IN-	DM-B
21	2	CTS	CS-A	3	IN+	CS-B
22	2	DSR	DM-A	3	RxC	RS-B
23	2	SG	TR-A	3	CS	RD-A
24	2	RLS	RR-A	3	CS	-
25	2	CS	-	3	RLS	RR-A
26	2	CS	RD-A	3	SG	TR-A
27	2	RxC	RS-B	3	DSR	DM-A
28	2	IN+	CS-B	3	CTS	CS-A
29	2	IN-	DM-B	3	RTS	RS-A
30	2	DTR	TR-B	3	RxD	RD-A
31	2	OUT+	RR-B	3	TxD	SD-A
32	2	OUT-	SG	3	TxC	SD-B

Appendix G IV1625 SCHEMATICS


P1
P2
P3
P4, P11
P6, P12
P8, P13
P7
P9
J1
J2
J3
J12
U1 'F245
U2 'F240
U3
U4 'F10
U5 'S38
U6 'G4
U7 'LS164
U8
U9 82S153
U10
U11 '375
U12 Z8530PC 4MHZ
U13
U14 'F74
U15 '161
U16 'LS00
R1 470
C8 10UF
C9 10UF
C10 10UF
C11, C12 10UF
RN1 2.2K
RN1-9
CR1
J11
J13
J16
J19
J14
J15
J17
J18
LIACK*
SIOIACK*
ENDWR*
BLOCK*
INT1*
INT2*
SYSCLK
DTACK*
IACKIN*
IACKOUT*
WRITE*
SYSRESET*
DS0*
IRQ7*
IRQ6*
IRQ5*
IRQ4*
IRQ3*
IRQ2*
IRQ1*
D00-1206-1.0
IV-1625-4SI0
1-31-85
3-31-86
1 OF 2


P2
P10, P14
2
1
3
5
7
9
11
13
15
17
19
21
23
25
26
J4
C16
C15
C14
C13
C12
C11
C10
C9
C7
C6
C5
C4
C3
C2
C8
C1

Appendix H IV1625 SIO MODULES

Ordering Information

SIO modules for the IV1625, generic part number IV103x, come in various flavors. The following table describes the currently available SIO modules.

TABLE H-1. SIO Modules

Part Number	Description
IV1030-C	RS-232 DCE
IV1030-T	RS-232 DTE
IV1031	RS-422 DCE
IV1032	RS-422 DTE
IV1033-C	Current Loop DCE
IV1033-T	Current Loop DTE

Note Modules that are indicated above with a '*' do not convey clock signals RxC and Txc between the Z8530 SCC and P2.

Cables

One cable is shipped with every SIO module ordered. DCE (DTE) cables have a DB-25 female (male) connector on one end and a .1X.1 16 pin connector on the other. The 16 pin connector fits on the P2 backplane. The same cable is used with RS-232 and Current Loop modules. RS-422 modules have a DB-37 connector replacing the DB-25.

IRONICS Incorporated

Computer Systems Division
798 Cascadilla Street
Ithaca, New York 14850
607-277-4060 Telex: 705-742