MULT I/O

Users Manual, rev 3

Preliminary

MORROW DESIGNS

MULT I/O

The Mult I/O is a general purpose S-100 utility card that combines all the board level features needed to form the heart of a powerful interrupt driven, real time, multi-user I/O system.

The board consists of the following I/O support sections:

- Three programmable serial I/O devices for communicating with data terminal equipment (terminals, printers, etc.) or data communications equipment (modems, computers, etc.).

- Three parallel ports (one input and two outputs) configured to plug directly into the 50 conductor ribbon cable of a Diablo compatible parallel 'Daisy Wheel' printer, and capable of being interfaced to most any parallel computer peripheral.

- A real time clock/calendar chip with provision for battery back-up, and the ability to cause timed interrupts at selectable intervals for multiprocessing and time-sharing applications.

- A programmable interrupt controller (PIC) capable of resolving eight levels of maskable, prioritized interrupts, five of which are asserted by the board itself.

- Four Kbytes of 2716 Eprom or four Kbytes of high speed static 6116 Ram (in the same sockets) or two Kbytes of each. Both Ram and Eprom are bank selectable and able to respond to all 24 S-100 address lines (extended addresses) as defined in the IEEE-696 buss specification.

The Mult I/O also provides a power-on-jump option which allows eight bytes of code to be executed from on-board Eprom during system power-on or reset.

The serial, parallel, clock and PIC devices on the Mult I/O are all I/O mapped - that is, they are accessed through switch selectable I/O port addresses. These devices may be programmed to request an interrupt of the PIC based on a multitude of status conditions. The 8259-A PIC can in turn issue to the CPU up to eight maskable, prioritized interrupt service routine vectors. As the sole system I/O card, one Mult I/O board can be used to support three terminals and a 'Daisy Wheel' printer while furnishing a real time, interrupt driven environment with all interrupt service routines optionally residing in on-board Ram and Eprom. Alternatively, up to four Mult I/O cards may be combined to accommodate as many as twelve terminals with full interrupt support.

I/O Addressing - General

All devices on the Mult I/O, even Ram and Eprom, are associated with an I/O port. In all, over 32 distinct I/O registers are used to control the many device functions available on the board. Yet the Mult I/O takes up only eight I/O port addresses. To understand how so many registers can be accessed through so few ports, it is helpful to think of the port addressing scheme of the Mult I/O as 'bank-selected I/O'. As the name suggests, this is analogous to conventional bank-selected memory schemes, in that, several 'banks' of registers are allowed to occupy the same area of consecutive I/O addresses while a dedicated I/O port is used to enable one 'bank' at a time.

The Mult I/O is divided into four I/O 'banks', hereafter called groups, with each group occupying the same seven I/O port addresses, namely, addresses BASE to BASE+6. Port address BASE+7 is the GROUP SELECT port, and is used to establish which of the four I/O groups will be active at any given time. This port is also used to control several other functions (enabling memory, interrupts and the printer port).

By sending a value between Ø and 3 to the GROUP SELECT port, the user enables the corresponding group of functions between BASE and BASE+6 for all subsequent I/O operations. To enable a different group the user must send a different group value to GROUP SELECT port BASE+7. While this port selection technique is extremely efficient in conserving I/O space, it does impose upon the user the responsibility of keeping track of which I/O group is currently active, and what the contents of the remainder of the GROUP SELECT port should be.

I/O Port addressing - Switch 7B

Switch 7B is used to determine the base address of the Mult I/O groups. Paddles 2 through 6 of switch 7B give the user control over address bits A7 to A3, respectively, so that the BASE port can be located at any eight byte boundary, starting at Ø and ending at port F8H. The relationship between paddle number and address bit is illustrated below:

DIP Switch 7B

Paddle Number	Address Bit
2	A7
3	A6
4	A5
5	A4
6	A3

'ON' = Ø and 'OFF' = 1

When a switch is closed, a low address bit compares true; when a switch is open, a high address bit compares true. When all conditions are true, the I/O is selected. For example, with all paddles 'ON' the Mult I/O would occupy I/O addresses Ø through 8. With all paddles 'OFF' it would occupy F8H through FFH.

Group Select Port BASE+7

Once the BASE address has been established by setting switches 2-6 of 7B, the addresses of all I/O functions on the Mult I/O are determined, as outlined in the I/O MAP on the following page. In order to gain access to a specific device function, however, the group number of that device must first be sent to I/O port BASE+7. The I/O group is selected by executing an output instruction to port BASE+7 with data bits Ø and 1 set as follows:

Group selection bits within BASE+7

Data Bit 1	Data Bit Ø	Group Number
Ø	Ø	Ø
Ø	1	1
1	Ø	2
1	1	3

NOTE: Beware of modifying the remaining data bits of this I/O port as these bits (D5-D2) control functions that have unique effects on the rest of the system (namely, bank selection, interrupt enable, and printer control).

As an example of using the GROUP SELECT port, suppose that we want the I/O space taken up by the Mult I/O to extend from 8ØH to 87H, and that we wish first to read ACE serial device #2 and subsequently to read "DAISY PORT" #Ø. I/O port BASE can be set to 8ØH by turning paddle 2 of switch 7B 'OFF' and paddles 3, 4, 5 and 6 'ON'. This would assign the GROUP SELECT port the I/O port number 87H. In order to read the data received buffer of the second ACE serial device (serial device number 2), the user must first set data bit 1 and clear data bit Ø of GROUP SELECT port 87H (to insure that I/O GROUP 2 is selected), and then read the desired data from port 8ØH (assuming the serial device has been properly initialized). To read the parallel 'Daisy' port, we would first switch to I/O GROUP Ø by clearing data bits 1 and Ø of port 87H, and then read the desired data from port 8ØH.

The important thing to remember here is that the function of I/O port 8ØH in our example changes from a serial device data register to a parallel device status register depending on the last byte we sent to the GROUP SELECT port, BASE+7.

The following page contains a general I/O map of the Mult I/O. For greater detail, such as data bit assignments, refer to the device data sheets included with this manual, and to the section of this manual dealing with the device in question.

I/O MAP

GROUP CONTROL I/O PORT BASE+7.
THIS IS A WRITE ONLY PORT, NO STATUS CAN BE READ.
TO SELECT AN I/O GROUP, OUTPUT TO PORT BASE+7
WITH DATA BITS Ø AND 1 SET AS FOLLOWS:

D1	DØ	GROUP#	DEVICE(S)
Ø	Ø	Ø	DAISY PORTS, 1991 CLOCK, 8259-A PIC
Ø	1	1	SERIAL PORT 1 (IC 2C, CABLE CONNECTOR J1)
1	Ø	2	SERIAL PORT 2 (IC 2B, CABLE CONNECTOR J2)
1	1	3	SERIAL PORT 3 (IC 2A, CABLE CONNECTOR J3)

ADDITIONAL CONTROL BITS OF BASE+7.
THESE BITS CONTROL BANK SELECT, INTERRUPT AND PRINTER ENABLES.

D5	D4	D3	D2	FUNCTION
Ø	Ø	Ø	1	IF 10B-2 IS 'OFF' RAM/EPROM IS ENABLED.
Ø	Ø	1	Ø	ENABLES THE INTERRUPT CONTROLLER (PIC).
Ø	1	Ø	Ø	TAKES PRINTER RESTORE (P4-13) LOW.
1	Ø	Ø	Ø	ENABLES ALL PRINTER OUTPUT LINES (P4).

ONCE A GROUP IS SELECTED, PORTS ARE ASSIGNED AS FOLLOWS:

GROUP Ø

	INPUT	OUTPUT
BASE	DAISYØ IN	DAISYØ OUT
BASE+1	not used	DAISY1 OUT
BASE+2	CLOCK IN	CLOCK OUT
BASE+3	not used	not used
BASE+4	8259-A AØ=Ø REGISTER	8259-A AØ=Ø REGISTER
BASE+5	8259-A AØ=1 REGISTER	8259-A AØ=1 REGISTER
BASE+6	not used	not used
BASE+7	not used	SELECT ALTERNATE GROUP

GROUPS 1, 2, & 3
(8250 ACE Serial I/O Ports)

	INPUT	OUTPUT
BASE	RECEIVE BUFFER/LSB BAUD	TRANSMIT BUFFER/LSB BAUD
BASE+1	INTERRUPT ENABLE/MSB BAUD	INTERRUPT ENABLE/MSB BAUD
BASE+2	INTERRUPT IDENTIFY	not used
BASE+3	LINE CONTROL REGISTER	LINE CONTROL REGISTER
BASE+4	MODEM CONTROL REGISTER	MODEM CONTROL REGISTER
BASE+5	LINE STATUS REGISTER	not used
BASE+6	MODEM STATUS REGISTER	not used
BASE+7	not used	SELECT ALTERNATE GROUP

Ram and Eprom - General

The Mult I/O is equipped to handle four Kbytes of high speed static Ram memory or four Kbytes of 2716 Eprom or two Kbytes of each. This memory occupies two sockets on the board, 5D and 6D (RØ and R1 respectively), and may be addressed to any 4K boundary within a 64K memory segment within a 16 Megabyte address space (see Extended Addressing). Ram/Eprom always functions as bank select memory (see Bank Selection), and is addressed as a single 4K unit.

No wait state is ever generated when addressing Mult I/O memory, which should be capable of running solid at up to 6 Mhz. If, however, the customer should have slow Rams or Eproms, he may either set his processor to generate wait states every cycle or M1 cycle (if possible) or he may order a special PLA from Morrow Designs which activates any number of wait states only on Eprom reads.

Addressing

Switches 3-6 of 1ØB control the addressing of the 4K block of Ram/Eprom within a given 64K memory segment. The paddles correspond to address bits as follows:

Memory Addressing
DIP Switch 1ØB

Address Bit	Paddle #
A15	3
A14	4
A13	5
A12	6

'ON' = Ø and 'OFF' = 1

Thus, to set Ram to begin at CØØØH, paddles 3 and 4 should be placed in the 'OFF' position, and paddles 5 and 6 should be placed in the 'ON' position. This will cause the left socket to occupy the address space from CØØØH to C7FFH. This same switch setting would place the right socket from C8ØØH to CFFFH. The following table gives the 16 possible settings of the Ram/Eprom address switch at 1ØB and the corresponding beginning and ending addresses of the left and right sockets.

Memory Address Settings
(Within a 64K segment)

A15 10B-3	A14 10B-4	A13 10B-5	A12 10B-6	Left Begin	Left End	Right Begin	Right End
Ø	Ø	Ø	Ø	ØØØØ	Ø7FF	Ø8ØØ	ØFFF
Ø	Ø	Ø	1	1ØØØ	17FF	18ØØ	1FFF
Ø	Ø	1	Ø	2ØØØ	27FF	28ØØ	2FFF
Ø	Ø	1	1	3ØØØ	37FF	38ØØ	3FFF
Ø	1	Ø	Ø	4ØØØ	47FF	48ØØ	4FFF
Ø	1	Ø	1	5ØØØ	57FF	58ØØ	5FFF
Ø	1	1	Ø	6ØØØ	67FF	68ØØ	6FFF
Ø	1	1	1	7ØØØ	77FF	78ØØ	7FFF
1	Ø	Ø	Ø	8ØØØ	87FF	88ØØ	8FFF
1	Ø	Ø	1	9ØØØ	97FF	98ØØ	9FFF
1	Ø	1	Ø	AØØØ	A7FF	A8ØØ	AFFF
1	Ø	1	1	BØØØ	B7FF	B8ØØ	BFFF
1	1	Ø	Ø	CØØØ	C7FF	C8ØØ	CFFF
1	1	Ø	1	DØØØ	D7FF	D8ØØ	DFFF
1	1	1	Ø	EØØØ	E7FF	E8ØØ	EFFF
1	1	1	1	FØØØ	F7FF	F8ØØ	FFFF

Extended Addressing

Extended addressing as applied to S-1ØØ memory devices is simply the ability of memory to decode more than 16 address bits. The 4K block of Ram/Eprom on the Mult I/O may be switched to decode 24 rather than 16 address lines - the extra 8 address lines being defined by the IEEE-696 buss specification. This extended addressing feature allows the Ram/Eprom on the Mult I/O to occupy any even 4K block within a 16 Megabyte address space. To enable the decoding circuitry, switch 1 of 1ØB must be placed in the 'OFF' position. Since most CPU boards currently in use do not generate address lines A23 - A16, many users will wish to disable this feature. This is done by setting switch 1 of 1ØB to the 'ON' position, and removing the IC at location 3D (25LS2521) from its socket.

With extended addressing enabled (switch 1 of 1ØB 'OFF', 3D installed), the DIP switch at location 2D determines the 64K segment in which the 4K of the Ram/Eprom will reside. The following table illustrates the settings of the switches at 2D and their corresponding extended address bits. The S-1ØØ buss pin numbers assigned by the IEEE-696 buss specification to these extended address bits are given in parentheses.

Extended Addressing
DIP Switch at 2D

Extended Address Bit	S-1ØØ Bus Pin #	DIP Switch 1D Paddle #
A23	(16)	1
A22	(17)	2
A21	(15)	3
A2Ø	(59)	4
A19	(61)	5
A18	(62)	6
A17	(63)	7
A16	(64)	8

DIP Switch 1ØB-1 must be 'OFF', and the IC at 3) installed to enable extended addressing.

'ON' = Ø and 'OFF' = 1

For example, to set Ram/Eprom to begin at 1ØCØØØH, set switch 1 of 1ØB 'OFF' to enable extended addressing, set the lower 16 bits (the CØØØ part of this address) of DIP switch 1ØB as per the instructions on the previous page, and set switch 4 of 2D 'OFF', and switches 1-3 and 5-8 'ON'. In this way, the left socket will respond to all memory accesses from 1ØCØØØH to 1ØC7FFH, while the right socket will be active from 1ØC8ØØH to 1ØCFFFH. When so addressed, Ram/Eprom will NOT respond to memory accesses to the area from ØCØØØH to ØCFFFH, and so would in effect be permanently disabled in any system incapable of generating extended addresses.

Bank Selection

The Ram/Eprom block on the Mult I/O is bank selectable memory - that is, an I/O instruction can cause the memory block to become enabled or disabled. In the case of the Mult I/O, when the host processor executes an output instruction to I/O port BASE+7 of any group, the Mult I/O board will examine data bit 2 and render its memory accessible or inaccessible according to whether it is high or low. If the bit is high and switch 1ØB-2 is 'ON', the Ram/Eprom will not respond to any further memory access unless:

1) The system is reset (and switch 1ØB-2 is 'ON'), or

2) A BASE+7 output occurs with data bit 2 low, or

3) Switch 1ØB-2 is turned 'OFF'.

Switch 1ØB-2 allows the user to determine whether or not Mult I/O Ram/Eprom will be selected after system power-up or reset. The setting of this switch also determines whether BASE+7 bit 2 will be active high or active low. If Switch 1ØB-2 is in the 'ON' position, then the Mult I/O Ram/Eprom bank will be enabled upon system power-up or reset, and BASE+7 bit 2 will have

to be low to enable memory, and high to disable it. If Switch 1ØB-2 is 'OFF', the Mult I/O Ram/Eprom bank will be disabled upon system power-up or reset, and will not be accessible until an output is made to port BASE+7 with bit 2 high. Thus 1ØB-2 'ON' causes the bank enable bit to be active low, and 'OFF' causes it to be active high.

PORT #	DATA BIT	I/O GROUP NUMBER
BASE+7	2	ANY

When disabled by bank de-selection, Mult I/O Ram/Eprom will 'disappear' from the bus, and so will not interfere with other system memory occupying an identical address. Therefore other bank select memory boards may be swapped in and out of memory along with Mult I/O Ram/Eprom. Of course, memory cards which are to be swapped in and out along with Mult I/O Ram/Eprom must themselves be capable of being disabled through some software mechanism. (Mult I/O memory is perhaps unique in that it can utilize extended address and bank select memory simultaneously.)

For example, if Mult I/O is set to begin at Port 48H, and if switch 2 of 1ØB is turned 'ON', then the Ram/Eprom can be enabled with the following routine:

```
;
; This subroutine does not modify the presently selected group.
;

bank:   lda     seldat      ;Recall old group select data.
        ori     4           ;Throw bank select bit high.
        sta     seldat      ;Save altered group select data.
        out     4f          ;Send to group select port
        ret                 ; with nothing but bank altered.

seldat: db      Ø
```

CAUTION!

The I/O devices (as opposed to the memory devices) on the Mult I/O board are not directly affected by bank selection or de-selection. However, since inadvertent selection or de-selection of Mult I/O Ram/Eprom could be disastrous, system programs using the Mult I/O should make some provision to avoid altering all BASE+7 devices when it is desired to alter only one of them. Inputting from port BASE+7 will NOT return the current status of selected group so some storage location should be used to hold current status, and routines which alter the group select should be sure to keep this storage location updated and to use its contents to mask out those bits which are not to be changed. The above routine, for example, sets BANK high without altering the previous selection.

Phantom Enable

Switch 1 of 7B is the Phantom Enable switch. When placed in the 'OFF' position, the Mult I/O will ignore 'Phantom/' (buss pin 67). In the 'ON' position, this switch causes the Ram/Eprom section of the Mult I/O board to become disabled and logically removed from the system buss whenever Phantom is active. Certain systems rely on the Phantom line to temporarily disable Ram memory in order to execute a special system start-up routine from Eprom. Once this routine is executed, the Eprom holding the routine vanishes and the Phantom line returns high to allow Ram memory to be accessed. Mult I/O memory is compatible with such a scheme.

Power-on Jump

Switch 10B-7 controls the power-on jump circuitry of the Mult I/O. When placed in the 'ON' position, this switch will cause the Mult I/O to force the host processor to execute the last 8 instructions of a Mult I/O Eprom.

For the power-on jump feature to be used, there must be at least one Eprom in either socket. Upon a reset or power-on, the board will point to the last eight bytes of Eprom to find a jump instruction to the users bootstrap routines. Switch 10B-8 determines which of the sockets will be used for the power-on jump code. The left socket (R0, at 5D) will be selected if this switch is 'ON' and the right socket (R1, at 6D) will be selected if it is 'OFF'.

Since the POJ circuitry of the Mult I/O can only jump to its own Eprom, this feature cannot be used unless:

1) The Eprom is enabled (10B switches 1 and 2, see above).

2) Switch 10B-8 is set to enable the correct Eprom socket.

3) The POJ switch is closed (10B-7 'ON').

4) There is an Eprom installed in the correct socket containing the desired jump routine in the last eight bytes.

Serial Ports

The Mult I/O has three 8250 programmable Asynchronous Communications Elements (ACE's) which may be connected to RS-232 devices via three 26 pin ribbon cable connectors. All three ACE's are configured as Data Communications Equipment (DCE) from the factory, and so may be connected with standard RS-232 terminals and printers. All may be re-strapped to act as Data Terminal Equipment (DTE), if they need to be connected to a modem or another computer. This eliminates the need to buy a custom cable for every new piece of equipment to be interfaced.

Each ACE has an I/O group dedicated to it - namely, groups 1, 2 and 3. The ACE's are completely programmable and so must be initialized in software before they can be used. Initialization includes setting the baud rate, word length, number of stop bits, parity, and interrupt conditions. Each ACE can be programmed to generate an interrupt in response to several conditions (data available, transmitter buffer empty, clear to send, etc.). The interrupt is sent directly to the Mult I/O PIC which may, in turn, pass it on to the host CPU. The interrupt handling routine may then interrogate the interrupt status register of the ACE responsible for generating the interrupt, and is thereby able to determine the precise cause of the interrupt.

The following chart gives the vital statistics of the ACE devices on the Mult I/O, including the location of the 8250 on the circuit board, the location of the 26 pin ribbon cable connector associated with each ACE, the I/O GROUP controlling each ACE, and the interrupt level assigned to each device by the 8259-A PIC.

ACE VITAL STATISTICS

	I/O GROUP #	26-pin connector	Board location	Interrupt Level
ACE # 1	1	P1	2C	3
ACE # 2	2	P2	2B	4
ACE # 3	3	P3	2A	5

P1 is the connector on the top left corner of the board; P2 and P3 are the next two connectors to the right of P1.

The pins on ribbon cable connectors P1-P3 are numbered so that the first 25 pins correspond exactly to the numbering of a standard DB-25 connector. Cables with flat ribbon cable connectors at one end and DB-25 connectors on the other are available off the shelf from many vendors.

Just below each serial connector is an array of seven jumper headers labeled J1, J2 and J3. They are configured with six slip on jumpers (wire wrap may be used for custom applications), and their main function is to select a DCE channel or a DTE channel. Configuration of these headers is illustrated below:

P1-P3 Connector Pinout (top view)

back

```
                14 15 16 17 18 18 2Ø 21 22 23 24 25 26
left                                                        right
                1  2  3  4  5  6  7  8  9  1Ø 11 12 13
```

front

J1, J2 and J3 - Serial Channel Set-up

As Data Communications Equipment (standard).

```
     A     B     C     D     E     F     G
     *     *     *     * -- *     * -- *
     |     |
     *     *     *     * -- *     * -- *
     H     I     J     K     L     M     N
```

As Data Terminal Equipment.

```
     A     B     C     D     E     F     G
     * -- *     *     *     *     *     *
                       |     |     |     |
     * -- *     *     *     *     *     *
     H     I     J     K     L     M     N
```

J1, J2 and J3 Pin Designations

EIA RS-232C pin #	J1, 2 or 3 pin	
2 - TXD	A	To cross reference a
3 - RXD	I	a software bit of the
4 - RTS	M	ACE with a physical
5 - CTS	G	hardware line at the
6 - DSR	E	connector, just find
8 - DCD	C*	the header pin letter
2Ø - DTR	K	associated the hardware
ACE pin & signal		
		or software signal
1Ø - sin	H	your interested in,
11 - sout	B	then look at the
32 - rts	F	header of the port
36 - cts	N	concerned. That
37 - dsr	L	letter should be
38 - rlsd	J*	jumpered to it's
33 - dtr	D	alternate in the table.

* These pins are hardwired together.

The pin assignment of the serial connector when the channel is configured as data terminal equipment (DTE - looks like a terminal) is described below:

Direction	Connector Pin #	RS-232 Function. (EIA Standard)	All names are with respect to DTE.
DCE -- DTE	1	Frame ground	
<-	2	Txmtd. data	
->	3	Rcvd. data	
<-	4*	Request to send	
->	5	Clear to send	
->	6	Data set ready	
--	7	Signal ground	
->	8**	Carrier Detect	
<-	20	Data term. rdy	

* Request To Send (RS-232 Pin 4) is implemented only on ACE # 1 and 2, and NOT on ACE # 3.

** Carrier Detect (RS-232 Pin 8 - ACE Received Line Signal Detect) is hardwired to be an INPUT to the board on all three channels, and so will sense a terminal's Carrier Detect line, but cannot be strapped to drive a modem's Carrier Detect line.

Also, Ring Indicator, RS-232 pin 22, is not implemented. Though this function has a dedicated line on the 8250 ACE and has its own status bit in the Modem Status Register, the 8250 RI pin (31) is tied high on the Mult I/O, and so sampling it would be meaningless.

Programming the 8250

An ACE device on the Mult I/O can be accessed only if its I/O GROUP is currently selected. Once a 1, 2 or 3 has been sent to GROUP SELECT port BASE+7, ACE device number 1, 2 or 3 can then be accessed. Each ACE contains internal 8 bit registers which occupy the first 7 I/O ports of the Mult I/O I/O space - BASE to BASE+6. The list below identifies all the internal registers of the 8250 and the I/O port address assigned to those registers by the Mult I/O:

I/O PORT	8250 ACE Register
BASE	If bit 7 of Line Control Register is 0, Receiver Buffer (Read) or Transmitter Holding Register (Write)
	If bit 7 of Line Control Register is 1, Baud Rate Divisor Latch low byte (R/W)
BASE+1	If bit 7 of Line Control Register is 0, Interrupt Enable Register (R/W)
	If bit 7 of Line Control Register is 1, Baud Rate Divisor Latch high byte (R/W)
BASE+2	Interrupt Identification Register (R/O)
BASE+3	Line Control Register (R/W)
BASE+4	MODEM Control Register (R/W)
BASE+5	Line Status Register (R/O)
BASE+6	MODEM Status Register (R/O)

The user should familiarize himself with the 8250 ACE data sheets provided with this documentation.

Baud Rate

The 8250's on the Mult I/O are completely programmable, including the baud rate. There is a 16 bit divisor latch within each ACE that divides the reference frequency into 65,535 possible rates, quite a few of which are precisely the standard EIA RS-232 transmission and reception baud rates. All ACE's have been hard wired so that the baud rate for data coming in is the same for data going out. The crystal used to provide the reference frequency for the three ACE devices is 1.8432 MHz. The data sheets give a broad sample of the divisors which must go into the Divisor Latch in order to generate the most common baud rates, and any baud rate may be generated from DC (this will inhibit all data transmission) up to 38,400 baud.

The formula for determining the divisor constant to produce a given baud rate is:

DIVISOR = 1.8432 M/(BAUD RATE X 16)

Although in most applications the user will simply look up the baud rate divisor in the data sheet table, there are instances when 'odd ball' baud rates may be useful - if, for example, an ACE is being used solely to generate interrupts at timed intervals based on the Transmitter Holding Register Empty interrupt (see Serial Device Interrupts).

Initialization

Though the reset pin (MR) of each 825Ø will be asserted during power-on or reset, no assumptions should be made about the contents of any 825Ø register unless that register has been previously initialized. Keep in mind that an on-board ACE cannot be accessed, far less initialized, unless its I/O group is selected. The Line Control, Modem Control, Interrupt Enable and Divisor Registers should be initialized before any data is transferred to or from an 825Ø.

The following three software routines are bare bones samples of how a Mult I/O ACE device could be driven in a CP/M* type environment. All of these routines adhere to CP/M* I/O protocol. The INIT routine sets up ACE # 1 to run at 96ØØ baud with an 8 bit word, no parity and 2 stop bits. The Interrupt Enable Register will be set to generate no interrupts, and the Modem Control/Status Registers will be ignored. This initialization would be appropriate for most RS-232 CRT terminals in a non-interrupt driven environment. Assume that the Mult I/O has been set to begin at 48H. The comments included with these routines may be used as a general flow analysis of ACE programming.

* CP/M is a trademark of Digital Research.

```
groupl  equ     1       ;code for first ACE (attached to J1)
base    equ     48h     ;base I/O address set by 7B 2-6
grpctl  equ     base+7  ;board group control port
dll     equ     base    ;ACE baud rate divisor (lsb)
dlm     equ     base+1  ;ACE baud rate divisor (msb)
ier     equ     base+1  ;ACE interrupt enable register
lcr     equ     base+3  ;ACE line control register
lsr     equ     base+5  ;ACE line status register
rbr     equ     base    ;ACE receiver buffer register
thr     equ     base    ;ACE transmitter holding register
dlab    equ     80h     ;divisor latch access bit
thre    equ     20h     ;line status register THRE bit
dr      equ     1       ;line status register DR bit
baudl   equ     12      ;divisor latch low byte - 9600 baud
baudh   equ     0       ;divisor latch high byte - 9600 baud
wls0    equ     1       ;word length select bit 0
wls1    equ     2       ;word length select bit 1
stb     equ     4       ;stop bit count - 2 stops
imask   equ     0       ;interrupt mask - disable all

;The following routine initializes the ACE as described above

init:   mvi     a,groupl ;set up desired I/O group.
        out     grpctl  ;select first serial device.
        mvi     a,dlab  ;open divisor latch access bit.
        out     lcr     ;base reg is now lsb baud rate reg.
        mvi     a,baudl ;low byte of baud rate divisor
        out     dll     ;into low baud rate register.
        mvi     a,baudh ;high byte of baud rate divisor
        out     dlm     ;into high baud rate register.
        mvi     a,wls0+wls1+stb    ;set up transmission format
                                   ;and close dlab.
        out     lcr     ;send to line control register.
        in      rbr     ;clear data ready flag in line status.
        mvi     a,imask ;interrupt mask set up.
        out     ier     ;base+1 now interruptmask - not baud.
        ret             ;end of initialization routine.

;The following routine will return in the accumulator any new
;character typed to ACE # 1

conin:  mvi     a,groupl ;select group one.
        out     grpctl
        mvi    a,wls0+wls1+stb  ;make sure dlab is clear.
        out     lcr
coninl: in      lsr     ;check line status register.
        ani     dr      ;any new data from terminal?
        jz      coninl  ;no, wait.
        in      rbr     ;get data.
        ani     7fh     ;strip parity.
        ret
```

```
;The following routine will send the character in Register C
;to ACE # 1

conout:  mvi     a,groupl ;select group one.
         out     grpctl
         mvi     a,wlsØ+wlsl+stb  ;make sure dlab is clear.
         out     lcr
conoutl: in      lsr      ;check line status register.
         ani     thre     ;is ACE ready to transmit?
         jz      conoutl  ;no, wait.
         mov     a,c      ;CP/M sends character in 'c'.
         out     thr      ;send to terminal.
         ret


;The following routine will return an FF in the Register A if ACE
;device # 1 has received a new character (i.e., DR is set in the
;ACE line status register).  Otherwise, return a Ø.

status:  mvi     a,groupl ;select group one.
         out     grpctl
         in      lsr      ;check line status register.
         ani     dr       ;data ready?
         rz               ;no.
         cma
         ret              ;yes.
```

In the above examples, it should be noted that the GROUP SELECT port is re-initialized at the beginning of every routine. This is done to insure against inadvertently sending serial I/O instructions to the wrong serial port or the clock, parallel ports or interrupt controller. It should be further noted that before accessing the ACE data register, the format word is again sent to the Line Control Register. This is done so that port BASE of GROUP 1 will be interpreted as a data port rather than as a divisor port. This guards against a situation such as losing access to the console device due to a careless reading of the Divisor Latch (from a monitor or front panel, for example) without subsequently clearing DLAB.

Serial Device Interrupts

The three 8250 ACE devices on the Mult I/O each have a dedicated interrupt request line on the 8259 PIC. The chart below describes the PIC interrupt level assigned to each ACE:

ACE Interrupt Assignments on 8259 PIC

Serial Device	PIC Interrupt Request Line
ACE # 1 (I/O Group 1)	IR3
ACE # 2 (I/O Group 2)	IR4
ACE # 3 (I/O Group 3)	IR5

ACE Interrupt Programming

As explained in the data sheet on the 8250, each ACE device can be programmed to generate an interrupt on any of four general conditions. These conditions are, in order of descending priority: Receiver Line Status, Received Data Available, Transmitter Holding Register Empty, and Modem Status. The Received Data Available and the Transmitter Holding Register Empty interrupts can be identified directly from the Interrupt ID Register of the source ACE. The remaining two interrupts must use the Interrupt ID Register to point to either the Receiver Line Status Register or the Modem Status Register. These two registers each have four interrupt flags which can be read to identify the source of an ACE generated interrupt. (The third interrupt of the Modem Status Register - The Trailing Edge of Ring Indicator, or TERI - is not supported by the Mult I/O, since the Ring Indicator line of each ACE is tied to +5V.) Because the 8250 prioritizes its interrupts, the Interrupt ID Register will 'freeze' the highest priority interrupt pending by ignoring all further interrupts until the previous interrupt has been serviced. For detailed information of the interrupt structure of the 8250 see the data sheets.

When using the 8250's ACE devices on the Mult I/O to generate interrupts, it is advisable to set the 8259-A PIC to operate in level mode, rather than edge mode. In edge mode, it is possible under certain circumstances for an ACE generated interrupt to be 'lost' - that is, to go unrecognized.

The Parallel "DAISY PORT"

General

The Mult I/O contains parallel I/O ports configured to accommodate a standard DIABLO type daisy wheel R/O printer. These ports are brought out to the 5Ø pin ribbon cable connector at P4 for easy attachment to a Diablo style printer. The pin assignments of P4 correspond exactly to those of an internal Diablo 5Ø conductor flat cable connector, so simply tying the Diablo to the Mult I/O via a ribbon cable with female sockets at either end is the only hardware requirement for interfacing the two devices.

Altogether, two latched output ports (plus an extra latched output bit) and one transparent input port are used to communicate with the Daisy Wheel printer. Of course, these ports may be used with practically any parallel device (e.g., a Centronics style printer or a keyboard) provided that the I/O lines are properly routed from the Mult I/O connector at P4 to the target device. This additional cabling burden is to be expected in parallel I/O interfacing, and so should not be considered a major disadvantage by those using the DAISY PORT with a non-Diablo parallel device.

The Mult I/O DAISY PORT occupies I/O ports BASE and BASE+1 plus a part of BASE+7 - all within I/O GROUP Ø. A single input line (BASE+Ø bit 5, or the Print Wheel Ready line when interfacing with a Daisy Wheel printer) is, after going to the DAISY PORT, inverted and then brought to IRQ 6 of the 8259-A interrupt controller, and so can be used to generate an interrupt whenever it goes to a low logic state. The eight input lines brought to DAISY PORT BASE are also pulled up to +5V through 18Ø Ohms (nominal), and so may be used with open-collector devices. These eight input lines are inverted by an input buffer, and so if left unconnected will appear to software to be low.

BASE+7 bit 5 enables all DAISY PORT output drivers. If this bit is low, DAISY PORT output lines controlled by I/O ports BASE, and BASE+1 will all remain at a high impedance state regardless of any software commands.

The parallel ports have no special facility for generating a strobe on output or latching a strobe on input. All data lines operate as levels, so strobes must be generated in software.

The following page depicts the parallel lines available on the Mult I/O, including the I/O port and bit number controlling each line and the function assigned to each line on a standard parallel Diablo type interface. Remember that these functions have no inherent meaning to the Mult I/O, which simply sees so many latches, and so do not preclude interfacing the Mult I/O with parallel devices other than Daisy Wheel printers.

DAISY PORT Signals and I/O Map

I/O GROUP Ø

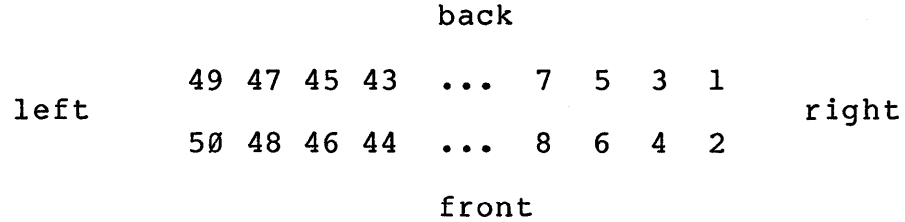
I/O Port	Data Bit	Mult I/O and Diablo Pin #	Diablo Function
Input BASE	Ø	4	End of Ribbon (-)
	1	3	Paper Out (-)
(these 8	2	5	Cover Open (-)
input lines	3	34	Paper Feed Ready (-)
pulled up	4	26	Carriage Ready (-)
to +5V by	5	27 *	Print Wheel Ready (-)
18Ø Ohms	6	12	Check (-)
& inverted)	7	28	Printer Ready (-)
Output BASE	Ø	37	Data Bit 1 (1) (-)
	1	36	Data Bit 2 (2) (-)
	2	39	Data Bit 3 (4) (-)
	3	33	Data Bit 4 (8) (-)
	4	4Ø	Data Bit 5 (16) (-)
	5	42	Data Bit 6 (32) (-)
	6	43	Data Bit 7 (64) (-)
	7	45	Data Bit 8 (128) (-)
Output BASE+1	Ø	46	Data Bit 9 (256) (-)
	1	1	Data Bit 1Ø (512) (-)
	2	9	Data Bit 11 (1Ø24) (-)
	3	1Ø	Data Bit 12 (2Ø48) (-)
	4	15	Paper Feed Strobe (-)
	5	17	Carriage Strobe (-)
	6	21	Print Wheel Strobe (-)
	7	23	Ribbon Lift (-)
Output BASE+7	4	13	Restore (-)

* In addition to being associated with bit 5 of Input Port Base, pin number 27 of P4 (the Diablo Print Wheel Ready line) is also connected through an inverter to Interrupt Request line 6 (pin 24) of the 8259-A PIC. Thus this line may be used to generate an interrupt whenever any external device brings it low (e.g., when the print wheel is ready).

The following lines on Mult I/O connector P4 are tied to ground as perscribed by the Diablo Interface: 2, 8, 11, 14, 18, 2Ø, 22, 25, 3Ø, 31, 32, 35, 38, 41, 44, 47. Line 24, defined by Diablo as Select/, is also grounded.

Line 48 of Mult I/O connector P4 is defined by Diablo as +5V (Reference Out). This line is not used by the Mult I/O board and is open. Also left open are lines 6, 7, 29, and 5Ø.

DAISY PORT P4 Connector Pinouts (top view)



Programming the DAISY PORT

As with all I/O devices on the Mult I/O, the user must be careful, when accessing the DAISY PORT, to initialize the correct I/O group - in this case, GROUP Ø. Once the proper I/O Group has been selected, all data sent from the CPU to the parallel ports will be latched. By latched, it is meant that the data sent to a parallel port will appear on the appropriate pins on the P4 connector, and will remain there until either different data is sent to the port in question or until DRIVER ENABLE (port BASE+7 bit 5) is taken low. When this occurs all 17 parallel output pins of connector P4 will enter a high impedance state.

The 8 input lines of the DAISY PORT are available to the CPU through an inverter, so that when an input instruction is directed at DAISY PORT Ø, the CPU will read the complement of whatever data is on the appropriate lines of connector P4 at the time the input instruction is executed. There is NO provision for strobing data into a parallel input 'buffer' for later examination after the data to be read has gone away.

The Mult I/O DAISY PORT inverts its input lines but does NOT invert its output lines. Daisy Wheel printers use negative logic, so that a low signal is taken as active. Thus to assert, or make active, any output line when talking to a Daisy Wheel printer, the software must put the line low. Input lines from a Daisy Wheel printer, on the other hand, are inverted in hardware, and so will appear to software to be active high.

Generating an Output Strobe

To generate an output strobe to any of the parallel output ports, it is necessary to use a software mask. The line to be strobed must be output three times in succession, changing state each time, while the data lines associated with the same port must be allowed to remain unchanged. For example, to send a positive going strobe (low-high-low) on bit 4 of port BASE+7 (Restore/) without changing the other 7 bits being sent from that port, the following routines could be used. Note that this happens to be the group select port we are strobing, but any parallel output bit may be used.

```
set:      mvi a,data     ;Set up data to be sent to port.
          out base       ;Data is latched, but not asserted until
                         ; the output ports are enabled.
                         ; and not strobed until Restore/ makes
                         ; a low-to-high transition.

enbl:     lda seldat     ;Get previous group select data.
          ori 3Ø         ;Parallel output ports enabled and
                         ; Restore/ (to be strobed) set low,
          ani Øfch       ; the group selected (Ø)
          out base+7     ; and sent to the group select port.

pulse:    ani Øefh       ;Take Restore/ high.
          out base+7     ;The data at the parallel port is sent.
          ori 1Øh        ;Bring Restore/ low again.
          out base+7     ;No other bits are affected.
          sta seldat
          ret

seldat    db Ø
```

The DAISY PORT and Interrupts

The Print Wheel Ready status line of the DAISY port (P4 connector pin 27, BASE input port bit 5) is brought through an inverter to Interrupt Request line 6 of the 8259-A PIC. The PIC can therefore generate an interrupt whenever this line goes to an active (i.e. logic low) state. To take full advantage of this interrupt option when interfacing with a Daisy Wheel printer, and to exploit the Diablo printer's ability to buffer motion commands, printer driver software should be written so that the Print Wheel Strobe (P4 pin 21, BASE output port bit 6) is not activated until all carriage positioning commands have first been sent to the printer. Print after space will execute significantly faster than space after print. When the Print Wheel Ready line goes active the printer should be able to accept another motion-then-print sequence.

A sample Diablo printer driver for the Mult I/O can be found in the Appendix of this manual.

Calendar Clock/Timed Interrupt Generator

The 1990C CMOS Clock/Calendar/Interrupt Timer at location 15D supports a multi-user environment by providing two functions:

1) A calendar clock accessible from software and able to run off a battery when power to the system is shut down, and

2) A timed interrupt generator able to provide interval interrupts, with four possible programmable interval lengths.

The clock uses six bits of output port BASE+2 in GROUP 0 for setting the time and the interrupt interval. The clock also uses input port BASE+2 to reset its timed interrupt line, which is tied through a latch to Interrupt Request Line 7 (the lowest priority interrupt) of the on-board 8259-A PIC. The chart below shows the Mult I/O ports and data bits used by the 1990, and indicates the correspondence between data bit and 1990 pin number/function.

1990 CALENDAR-CLOCK I/O MAP

I/O Port BASE+2	BASE+2 Bit #	1990 Pin # & mnemonic	1990 Function
Input to CPU	0	9 - Data Out	Output of 40 bit shift register
Output from CPU	0	6 - Data In	Input of 40 bit shift register
	1	8 - Clk	Shift clock for 40 bit register
	2	3 - C0	Command input bit 0
	3	2 - C1	Command input bit 1
	4	1 - C2	Command input bit 2
	5	4 - STB	Strobe input

Clock Programming

The data sheets on the 1990 chip should be studied before attempting to program this device. The 1990 stores the time of day, day of week, and month of year in an internal 40 bit shift register which is accessible to the Mult I/O user through bit 0 of I/O port BASE+2 of GROUP 0. Commands to set or read time must be strobed into this port using bit 5 as the strobe bit, and the 40 bits of time data must be clocked in or out using bit 1 as the clock bit. The format of this internal 40 bit shift register is seven four-bit binary coded decimal nibbles and, for the month of the year, one hex nibble. The 40 bit shift register is a FIFO - first in, first out - and begins with the least significant bit. Thus the first bit in or out of the 40 bit FIFO is always the least significant bit of the single seconds nibble, and the last bit out is always the most significant bit of the month of the year nibble. Note in the following table how each individual nibble seems to be coded backwards.

Time Format of the 1990 40 Bit FIFO

(Bits are given in the order they should be written, and in the order they will be read.)

Bits 1 to 8 - Seconds (Ø to 59)

	seconds units				tens of seconds			
1990 bits	1	2	3	4	5	6	7	8
	lsb			msb	lsb			msb

example: 38 seconds would be stored as follows:

1990 bits	1	2	3	4	5	6	7	8
logic level	Ø	Ø	Ø	1	1	1	Ø	Ø
interpretation	8				3			

Bits 9 to 16 - Minutes (Ø to 59)

	minutes units				tens of minutes			
1990 bits	9	1Ø	11	12	13	14	15	16
	lsb			msb	lsb			msb

example: 41 minutes would be stored as follows:

1990 bits	9	1Ø	11	12	13	14	15	16
logic level	1	Ø	Ø	Ø	Ø	Ø	1	Ø
interpretation	1				4			

Bits 17 to 24 - Hours (Ø to 23)

	hours units				tens of hours			
1990 bits	17	18	19	2Ø	21	22	23	24
	lsb			msb	lsb			msb

example: 11 o'clock p.m. (23ØØ hours) would be stored as follows:

1990 bits	17	18	19	2Ø	21	22	23	24
logic level	1	1	Ø	Ø	Ø	1	Ø	Ø
interpretation	3				2			

Time Format of the 1990 40 Bit FIFO - continued

Bits 25 to 32 - Day of Month (1 to 31)

	day units				tens of days			
1990 bits	25	26	27	28	29	30	31	32
	lsb			msb	lsb			msb

example: the 14th of the month

1990 bits	25	26	27	28	29	30	31	32
logic level	0	0	1	0	1	0	0	0
interpretation			4				1	

Bits 33 to 36 - Day of the Week (0 to 6)

1990 bits 33 34 35 36
lsb msb garbage bit

Sunday = 0
Monday = 1
Tuesday = 2
. . .
Saturday = 6

example: Thursday

1990 bits	33	34	35	36
logic level	0	0	1	0
interpretation			4	

Bits 37 to 40 - Month of the Year (0 to B Hex)

1990 bits 37 38 39 40
lsb msb garbage bit

January = 0
February = 1
March = 2
. . .
November = A Hex
December = B Hex

example: July

1990 bits	37	38	39	40
logic level	1	1	1	0
interpretation			7	

Idiosyncracies of the 199Ø Calendar Clock

Once the 4Ø bit shift register of the 199Ø has been set with the desired time and date, it will automatically increment the time and date for later reference. Note, however, that the 199Ø considers all months to have 31 days, so September, April, June and November - and certainly February - require a special update at the end of the month in order not to throw the calendar off.

Strobe and Clock Timing

The 199Ø is not capable of reading or writing serial data fast enough to keep up with the CPU unless the clock and strobe bits are prolonged for about 7ØØ micro-seconds. This can be easily accomplished in software.

Software Flow for Writing the Time/Date to the 199Ø

Writing the time to the 199Ø requires a four step procedure:

1: Select I/O GROUP Ø of the Mult I/O.

2: Strobe the Register Shift Command to I/O port BASE+2

3: Clock forty consecutive bits to the Data In pin of the 199Ø. Each bit should be sent via three output instructions to I/O port BASE+2, with suitable delays in between, in which the Data Bit (bit Ø) stays the same, the Strobe Bit (bit 5) stays low, and the Clock Bit (bit 1) is first high, then low, then high again (see note below).

4: Strobe the Set Time Command to I/O port BASE+2.

Software Flow for Reading the Time/Date from the 199Ø

Reading the time also requires a four step procedure:

1: Select I/O GROUP Ø of the Mult I/O.

2: Strobe the Read Time Command to I/O port BASE+2

3: Strobe the Register Shift Command to I/O port BASE+2

4: Clock forty consecutive bits from the Data Out pin of the 199Ø. Each bit should be read via three input instructions from I/O port BASE+2, with suitable delays in between, in which the Strobe Bit (bit 5) stays low, and the Clock Bit (bit 1) is first high, then low, the high again (see note below).

The appendix contains a source listing of program which can write the time to the clock or read it back.

It is a good idea to have interrupts disabled when writing to or reading from the clock, since a lengthy interrupt service routine could cause the data read or written to be inaccurate.

The Timed Interrupt Generator

In addition to being a calendar/clock, the 199Ø is capable of generating interrupts at timed intervals. The interrupts generated by the 199Ø are routed to Interrupt Request number 7 of the 8259-A PIC. In order for these interrupts to be received properly, the PIC must be set to operate in level, rather than edge, mode. Three interval times are available and are selected under software control. The intervals are:

1) once every Ø.488 milliseconds, or 2Ø48 interrupts per second;
2) once every 3.9 milliseconds, or 256 interrupts per second;
3) once every 15 milliseconds, or 64 interrupts per second;
4) once every 3Ø milliseconds, or 32 interrupts per second.

Generating a Timed Interrupt

As indicated in the data sheet on the 199Ø, the TP (Timed Pulse) output, which is the source of the 199Ø interrupts, can be programmed to oscillate with a 5Ø% duty cycle at one of three frequencies. These frequencies are selected by strobing the appropriate data into Mult I/O port BASE+2.

An example of how to set the Timed Pulse to the desired interval:

```
tprate:   lda seldat     ;Get previous group select data.
          ani Øfch       ;Select group Ø without changing the
          out base+7     ; remaining control bits.
          sta seldat     ;Save new group select data.

          mvi a,1ch      ;In this case, the test mode which
                         ; generates a 32Hz interval.
                         ;Other rate values:
                         ;1Øh = 64Hz, 14h = 256Hz, 18h = 2Ø48Hz.
stb:      out base+2     ;Send to the clock chip data latch.
          xri 2Øh        ;Set the STB high, retaining rate data.
          out base+2     ;Data is strobed on the positive edge.
          ret

seldat:   db   Ø
```

Port data	TP Frequency	Time Between Interrupts
1Ch	32 Hz	3Ø msec.
1Øh	64 Hz	15 msec.
14h	256 Hz	3.9 msec.
18h	2,Ø48 Hz	Ø.488 msec.

Clearing the Timed Interrupts

An input instruction directed at I/O port BASE+2 will clear the interrupt request generated by the 199Ø. This clears the flip-flop through which the 199Ø TP output is latched (and thus held at a constant level) before reaching the 8259-A PIC.

Generating Interrupts at Non-standard Intervals

If the interval selection available on the 199Ø does not fit the user's application, a broader selection is possible by using an on-board 825Ø ACE - simply by programming the ACE to generate an interrupt whenever the Transmitter Buffer is empty.

Battery Back-up for the 199Ø Clock

The Mult I/O allows for the clock to retain it's information when the system power is turned off. The three prong plug at 14D (P6), is a non-polarized power supply connector for a 3 volt battery pack with the two outside pins grounded and the center +3 volts. Two 1.5 batteries, (AA cells for example) will keep the clock running without system power for very close to their shelf life, due to the very low current drain of the CMOS 199ØC chip. The supply voltage of the 199ØC is unlike other CMOS in that it must be between 1.8 and 3.6 volts, therefore, different batteries (say, 9 volt) must be dropped by an appropriate resistor which may be installed at 15D.

The 8259-A PIC

General

Interrupts are a method of directing the CPU to execute specified segments of code in response to events, rather than as a result of the 'natural' progression of the program counter through a sequential program. A great advantage of an 'interrupt driven' system as compared with a non-interrupt system is that the CPU is freed from spending much of its execution time merely 'polling' I/O devices - i.e., repeatedly reading status registers to determine whether a byte of data is ready to be processed. With interrupts, the CPU can do some useful task until an I/O device 'tells' the CPU, without being asked, of an important change in status, and then vectors the CPU off to an appropriate service routine. The improvement in CPU performance made possible through interrupts is particularly evident in a multi-user system, where several users share a single CPU. In fact, a multi-user environment using currently available microprocessors but without the use of interrupts is scarcely feasible.

The advantage of interrupts, then, is speed and efficiency, realized by an increase in throughput. The penalty is a corresponding increase in the complexity of both hardware and software. The Mult I/O board is designed to provide all the hardware necessary to implement on the S-1ØØ buss a powerful interrupt driven system, while at the same time minimizing the software burden normally associated with interrupts.

Interface Between PIC and Other Devices

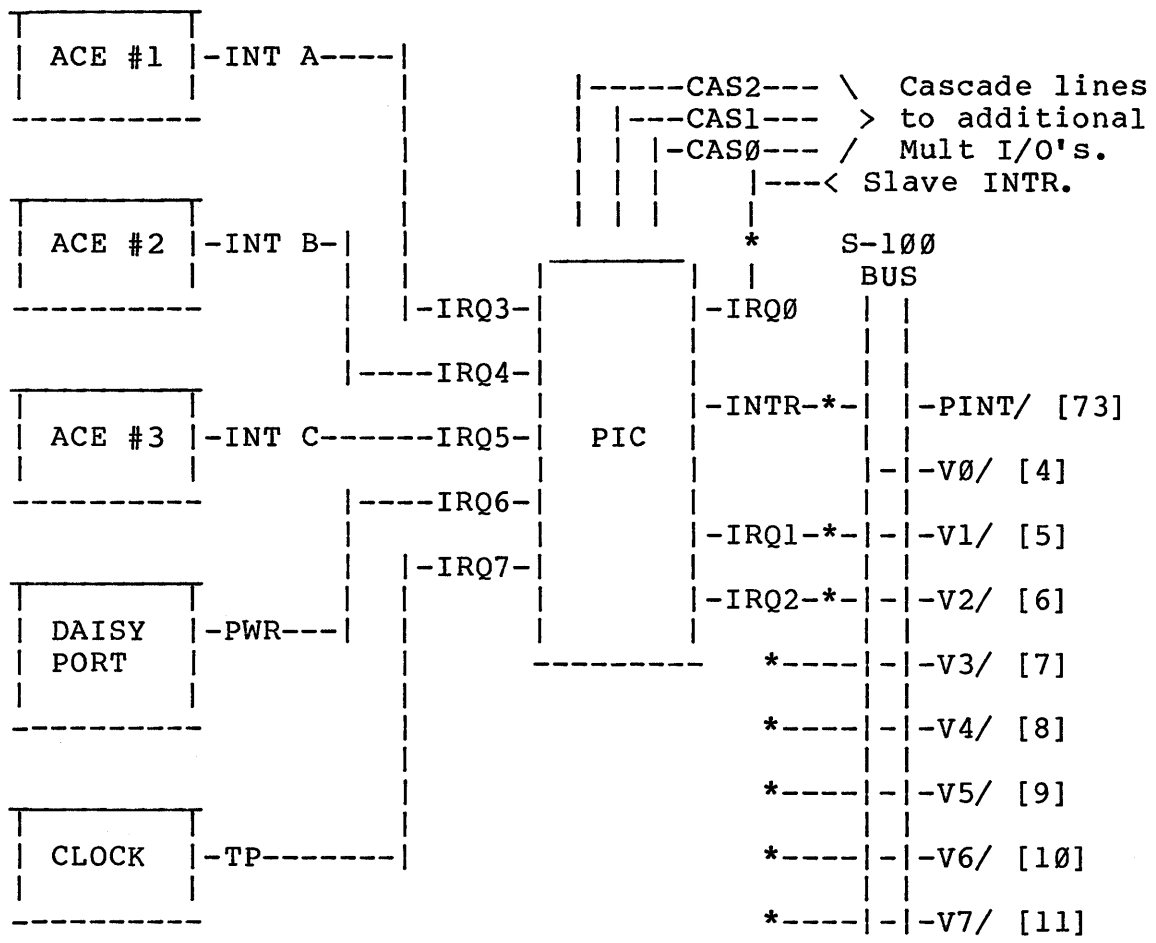
The 8259-A Programmable Interrupt Controller (PIC), is the heart of the Mult I/O. Interrupts generated by the three on-board ACE serial devices, the Print Wheel Ready line of the parallel ports, and the Timed Pulse (TP) signal of the Real Time Clock, are hard wired to the PIC Interrupt Request lines 3 to 7.

The on-board interrupt controller may also monitor any three vectored interrupt lines and can assert either the generalized interrupt request line (PINT/) or any vectored interrupt line (selected by the user). Thus interrupts generated from three off-board devices may be routed to the Mult I/O PIC using the vectored interrupt lines (master mode), or the Mult I/O PIC can send its interrupt requests over the vectored interrupt lines to another interrupt controller (slave mode).

Multiple Mult I/O boards may be cascaded in this way. A four pin connector next to the PIC at 11A is used to cascade multiple Mult I/O interrupt controllers, allowing up to three additional boards (slaves) to be installed in the buss. This facilitates 12 serial ports, four parallel printer ports, four interrupt timers and five active vectored interrupt lines. The cascade cable is used to significantly reduce interrupt software overhead.

This illustration shows the interface of the PIC with on-board I/O devices and with the S-100 bus. In this example, the PIC is set up as master with one slave.

Interface of PIC With On-board and Off-board I/O



* Asterisks represent user installable jumpers.

Configuring the PIC

The 8259-A PIC can be used in two general modes - master or slave. In master mode, the PIC not only generates an interrupt but also provides a CALL instruction (and the address of a service routine jump table) to the CPU. In slave mode, a PIC does not request an interrupt of the CPU, but of the master which asserts the CALL instruction and not the address. The slave then supplies the CPU with the address of the jump table.

In order to determine the PIC jumper configuration most appropriate for a particular application, it is necessary to understand how the Mult I/O card uses the 8259-A PIC to generate

interrupts. When used as the system master interrupt controller, the PIC will issue its interrupt vectors to the CPU through a four step sequence:

1) The PIC will assert its INT line in response to a valid interrupt request on one of its eight IRQ lines. As a master, J5-B is jumpered to the PINT/ line of the S-1ØØ buss (at J5) which will be driven low, requesting an interrupt of the CPU.

2) If interrupts are enabled (an EI instruction has been executed and has not been cancelled by a subsequent DI instruction), the CPU will complete its current instruction and assert the INTA (Interrupt Acknowledge) buss line during the next M1 (instruction fetch) cycle. The address lines asserted by the CPU at this time will point to the next byte of data to be fetched. Fortunately, standard memory boards will not respond to any read cycle occuring when INTA is asserted.

3) The Mult I/O board will use the INTA pulse to clock out of the PIC the first byte of a CALL instruction (CD) over the S-1ØØ Data In lines. Because of the assertion of the INTA line, no other device should be driving the Data In lines at this time.

4) The CPU will read this CALL instruction and attempt to read two additional bytes of data to form a 16 bit call address. Most CPU's will not assert INTA during either of these subsequent fetches, so that these last two read cycles will appear to any memory board to be standard memory read cycles, and the address lines normally asserted by the CPU will again be those that would have been asserted had there been no interrupt. The Mult I/O board will allow the PIC to send the two address bytes to the CPU during these two fetches.

If configured as a master, the Mult I/O will generate the address of its own Eprom (which is temporarily disabled) for three consecutive read cycles whenever it receives an INTA. The Address Disable buss line will be asserted by the Mult I/O board whenever the PIC is generating its interrupt vectors. This places the S-1ØØ address lines in a known state during an interrupt sequence. As long as no system memory occupies the same address space as Mult I/O Eprom, no memory conflict will occur.

Jumper Area J5

If the Mult I/O is to be used as the master interrupt controller, The interrupt request line of the PIC (J5-B) must be connected to the S-1ØØ buss interrupt line PINT/ (in J5 area).

If the Mult I/O is to be used as a slave, the interrupt line from the PIC (J5-B) should be routed to one of the eight vectored interrupt request lines (VIØ-7) to be sensed by the master. The vectored interrupt line to be used depends on which line the master expects to see a slave. J5 pads C,D and E are the three available inputs to the PIC. They are etched to VI Ø,1 and 2

respectively and may be altered by the user. In order to prevent a slave from responding to it's own interrupt requests, the VI line that the slave uses should be cut from it's own input, or the slave must be prevented from ever responding to that input by the software.

Disabling Mult I/O Interrupts

To disable all interrupts from the Mult I/O card, J4 (at 12C) should have A - B jumpered and B - C open. This prevents the address buss from being asserted by the Mult I/O (thus, the CALL instruction and address are never found). This does not, however, prevent any Mult I/O from asserting it's interrupt request on PINT/ or the vectored interrupt lines. This is accomplished in hardware by simply cutting J5 pads A and B away from any other pad in the J5 area. A not so simple approach in software is to NEVER set bit 3 of BASE+7 to a high state, or to be safe, initialize the master's interrupt controller to ignore all requests by setting all bits of OCW1 high.

Summary of PIC Related Jumpers.

Jumper Area J5

```
                                        A = Single slave int.
                                        B = Int. req. from PIC.
A     B     C     D     E               C = PIC input IRQ Ø.
*     *     *     *     *               D = PIC input IRQ 1.
            |     |     |               E = PIC input IRQ 2.
            |     |     |
            *     *     *     *     *     *     *     *     *
      VI  -  Ø     1     2     3     4     5     6     7     PINT/
```

When the cascade cable is installed, [A] may be used to route a single slave's interrupt request [B] to one of the master PIC inputs. In this way, none of the S-1ØØ vectored interrupt lines need be used. This only applies to a master and one slave, additional Mult I/O slaves must send their interrupt requests down any of the vectored interrupt lines of the buss.

If, for example, the system were to contain a master with only one slave on IRQ2, the slave would have [B] connected to [A] and the master would have [A] connected to [E] and [E] cut away from VI2. The master (in order to BE a master) must also have [B] connected to PINT/. The PIC inputs [C] and [D] should only be connected on one of the boards so that the two controllers don't respond to the same VI line.

A system containing four Mult I/O boards (one master and three slaves) is similar in concept. Here is an example set-up.

We will use the three VI lines (Ø, 1 and 2) to send slave interrupt requests to the master. These inputs (C, D and E) must

be cut from all PIC's except the master to prevent slaves from responding to other slave's, or their own, vectored interrupt requests. Each slave should then have it's interrupt assertion line [B] routed to a distinct VI line (to give them names, slave zero's interrupt line would be connected to VIØ, slave one's to VI1 and slave two's to VI2). The only jumper on the master then, is [B] to PINT/ since the VI lines are already strapped to the master PIC inputs.

Jumper J4

A - * - Ground

B - * - To address control.

C - * - From Interrupt logic.

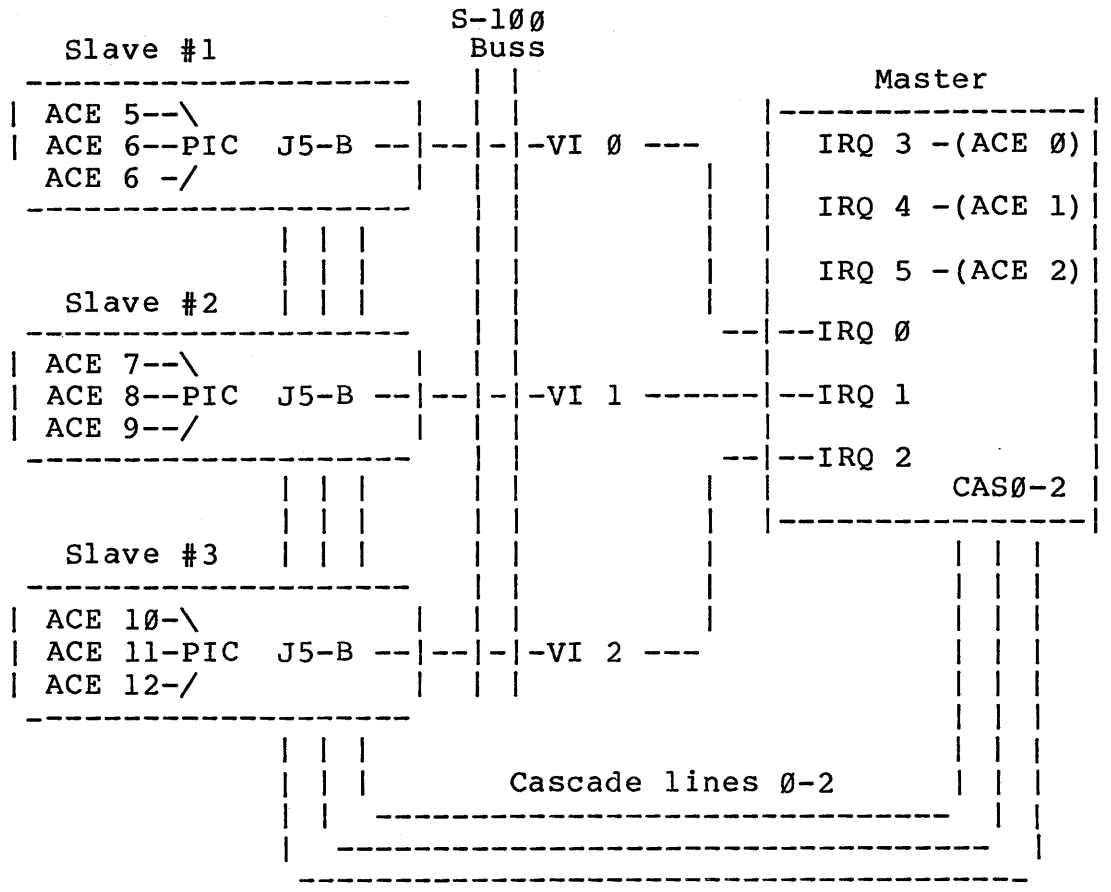
Connect B to C when using on-board 8259-A PIC as master or slave interrupt controller. Connect A to B when the 8259-A is not to be allowed to generate interrupt vectors. Jumper J4 determines whether or not the Mult I/O may issue vector information (a CALL instruction or address) in response to the CPU's Interrupt Acknowledge line.

Connector P5 - the Cascade Lines

*	*	*	*
|	|	|	|
To/From Area J5-A	To PIC CAS 2	To PIC CAS 1	To PIC CAS Ø

Using Several Mult I/O Boards in Master/Slave Configuration

If multiple Mult I/O boards are to be used in a system, one board should be configured as the master interrupt controller and the others as slaves. The slaves may communicate with the master through the vectored interrupt lines VØ, V1, and V2. A four board interrupt system is illustrated below:



In the configuration above, the master Mult I/O board should have it's interrupt request pad (J5-B) connected to the PINT/ line of the buss, while slaves should have this pad jumpered to the appropriate VI lines. Slaves should also have all three board traces beneath the pads marked C, D and E in J5 cut open.

Programming the PIC

Before attempting to program the 8259-A PIC the user should become familiar with the data sheet included with this documentation. The PIC monitors eight interrupt request lines (IRQ 0 to IRQ 7). These IRQ lines are maskable and prioritized. By maskable it is meant that, through software, it is possible to determine which if any of the IRQ lines will be monitored by the PIC at any given time. By prioritized it is meant that in case more than one unmasked IRQ line is asserted at a given moment, the one with the highest priority (normally corresponding to the one with the lowest IRQ number) will be serviced first. The conditions under which various devices will request service from the PIC are, in the cases of the ACE serial devices and the Real Time Clock, determined in software when the devices in question are initialized. Refer to the appropriate section of this manual to see how these I/O devices are initialized.

The PIC services valid interrupt requests which appear on its IRQ lines by forcing the CPU to read a three byte CALL instruction. The CALL instruction directs the CPU to a jump table located somewhere within either a 32 byte or a 64 byte block of system memory. Each IRQ line is associated with a unique location, or interrupt vector, within this block of system memory. IRQ Ø is dedicated to the first location of the block, and each succeeding IRQ line controls a vector either 4 or 8 bytes away from the preceeding one. The user determines the address of the first vector (limited to an even 32 or 64 byte boundary) and the space, whether four or eight bytes, between each successive vector. Below is an I/O map of the PIC and a list of the devices associated with each IRQ line.

The PIC can be accessed through Mult I/O ports BASE+4 and BASE+5 of GROUP Ø. The numerous registers of the PIC are accessed through only these two I/O ports, since context plays a large part in selecting a specific PIC register. The PIC pin AØ equals Ø when addressing port BASE+4, and 1 when addressing BASE+5.

PIC I/O Map and I/O Device Priorities

Priority	PIC Address	Mult I/O Port Address	
	AØ=Ø	GROUP Ø, BASE+4	
	AØ=1	GROUP Ø, BASE+5	
Highest	IRQ Ø	Vectored Interrupt Ø \	S-1ØØ
	IRQ 1	Vectored Interrupt 1 > -	Buss
	IRQ 2	Vectored Interrupt 2 /	Lines
	IRQ 3	ACE Serial Device 1	
	IRQ 4	ACE Serial Device 2	
	IRQ 5	ACE Serial Device 3	
	IRQ 6	DAISY PORT P.W.R. line	
Lowest	IRQ 7	Real Time Clock TP line	

Initializing the PIC

Only four PIC registers need be initialized in order to implement a powerful interrupt system. More complicated initialization schemes are left up to the imagination of the user. The recommended use of this board requires that three Initialization Command Words (ICW's), and one Operation Control Word (OCW) be written during the PIC initialization sequence.

ICW1 and ICW2

ICW1 and ICW2 are used 1) to set four flags which determine certain selectable operational characteristics of the PIC, and 2) to set the eight vector addresses which the PIC will cause the CPU to CALL when the PIC receives one of the eight possible interrupts. The four flags are set through bits Ø to 3 of ICW1, and the vector addresses are set through bits 5 to 7 of ICW1 and

through all eight bits of ICW2. ICW1 is accessed by sending the proper flag and vector information to Mult I/O port BASE+4 of GROUP Ø while data bit 4 is set. ICW2 is accessed by sending the remainder of the vector information to Mult I/O port BASE+5 of GROUP Ø immediately after accessing ICW1. The only way ICW2 is distinguished by the PIC from OCW1 is by the fact that ICW1 has just been accessed - that is, by context.

The flag bits set in ICW1 have the following functions:

ICW1 FLAG BITS

IC4 - (data Ø) This bit determines whether or not it will be necessary to initialize ICW4. It is, on the Mult I/O, so this bit should be set high.

SNGL - (data 1) This bit is used to inform the PIC of other interrupt controllers in the system. If this bit is high, ICW3 is not needed as there this is the only PIC in the system. See section on Configuring the PIC as a slave.

ADI - (data 2) This bit determines the CALL address interval. If cleared low, the CALL vectors generated by the PIC will point to memory locations seperated by eight bytes. If set high, the vectors will point to memory locations separated by four bytes.

LTIM - (data 3) This bit determines whether the IRQ lines of the PIC will be edge triggered or level triggered. For the Mult I/O board to function properly, this bit should be set, or put to a high, thus selecting the level mode.

Address Initialization - ICW1 & ICW2

In addition to setting the four flag bits in ICW1, the user must initialize the interrupt vector address of the PIC. The address where the PIC will vector the CPU during the IRQØ interrupt is the only address that must be programmed, since the other seven addresses must follow automatically at intervals of either four bytes (if ADI=1) or eight bytes (if ADI=Ø). If ADI=1, then bits 5 through 7 of ICW1 must be set to match address bits 5 through 7 respectively of the IRQØ vector. If ADI=Ø , then bits 6 and 7 of ICW1 must be set to match address bits 6 and 7 of the IRQØ vector. In both cases, ICW2 must be set to match the high byte of the IRQØ vector. Note that the memory address block pointed to by the PIC's CALL instructions must begin on an even 32 byte boundary if ADI=1 (an address divisible by 2ØH), and on an even 64 byte boundary if ADI=Ø (an address divisible by 4ØH).

Sample Initialization of ICW1 and ICW2

To initialize the PIC to operate as a single controller, in level triggered mode with an address interval of four, and to cause the PIC to vector the CPU to location ØC18ØH upon receipt of IRQØ, the following sequence must be followed:

1. Set bit 3 of port BASE+7 and select GROUP Ø to enable the controller.

2. Send a 9Fh to port BASE+4 (ICW1 since data bit 4 is set).

3. Send aC1h to port BASE+5 (ICW2 since last output was to ICW1).

4. Send a ØCh to port BASE+5 (ICW4. ICW3 was skipped because SNGL was high indicating a single controller). This sets the buffered mode (bit 3) turning pin 16 of the PIC into an output instead of an input. The input function of this pin is to tell a PIC whether it is a master or slave. With this pin now being used to gate data through the Mult I/O buffers, the determination must be made in software. Bit 2 of ICW4 being high sets the master buffered mode and when low sets the slave buffered mode.

The above sequence will cause the PIC to issue CALL's to location C18Øh as response to a valid IRQØ, to C184h for IRQ1, to C188h for IRQ2, and so on up to location C19Ch for IRQ7. Thus is ACE #1 generated a valid interrupt, the PIC would issue a CALL to location C18Ch.

OCW1 - The Interrupt Mask

After ICW1 and ICW2 have been initialized, the port address occupied by ICW2 becomes dedicated to OCW1, which controls the interrupt mask register of the PIC. The use of this port is quite simple. The IRQ lines Ø through 7 are made to correspond to data bits Ø through 7 respectively of this register. An OUT instruction to this register with any data bit set, or high, will

mask out the corresponding IRQ line, so that the PIC will ignore that line until the appropriate mask bit is re-enabled. For example, to allow the PIC to monitor the three Mult I/O ACE devices and to ignore all other IRQ lines, one would send a C7H to port BASE+5 of GROUP Ø. This would enable IRQ lines 3, 4 and 5 and disable the other four lines. If this register is not initialized, all IRQ lines are cleared and, hence, enabled. The correspondence between OCW1 data bits and Mult I/O devices is shown below:

OCW1 Mask Bits and Corresponding Devices/Lines

data bit	IRQ Line	Physical Device	Priority
Ø	Ø	VØ (S-1ØØ pin 4) *	Ø
1	1	V1 (S-1ØØ pin 5) *	1
2	2	V2 (S-1ØØ pin 6) *	2
3	3	825Ø ACE # 1	3
4	4	825Ø ACE # 2	4
5	5	825Ø ACE # 3	5
6	6	DAISY PORT PWR line	6
7	7	RT Clock TP line	7

* These connections may be altered by the user.

Note that in order to prevent ANY interrupt from being generated by the PIC, simply send an FFh to OCW1. Remember that Priority Level Ø is the highest priority level, meaning that an IRQØ interrupt takes precedence over, say, an IRQ3 interrupt.

Sending EOI (End of Interrupt) to OCW2

It is recommended that the 8259-A PIC on the Mult I/O be initialized with the ICW4 and LTIM flags high in ICW1. This means that the PIC should be operated in the level mode and that the user must terminate all Interrupt Service Routines (ISR's) by sending the PIC an End of Interrupt command (EOI). This means that once the PIC generates an interrupt, it will recognize no further interrupts until it receives an EOI command. The EOI command can be sent by sending a 2ØH to I/O port BASE+4 of GROUP Ø. This port will be taken as OCW2 by the PIC.

The Interrupt Status Register - OCW3

Occasionally it is desirable to read the Interrupt Status Register (ISR) of a PIC in order to determine the cause of an interrupt. First, GROUP Ø must be selected for the Mult I/O board whose PIC is to be examined. Then a ØBH must be sent to port BASE+4 of the Mult I/O board in question. Finally, that same port must be read. The data thus read will correspond to the priority levels being serviced by the PIC. These bits correspond to devices and IRQ lines as indicated in the above illustration on the Interrupt Mask Register.

Interrupt Software Using the 8259-A PIC

A typical interrupt service routine used with the Mult I/O PIC should perform the following four functions:

1. Service the device causing the interrupt.

2. Send an EI instruction to the CPU. This is necessary because interrupts are automatically disabled whenever an interrupt is received, so failure to terminate an ISR (Interrupt Service Routine) with an EI instruction would prevent any further interrupts from being serviced. Once an EI instruction is executed, higher priority interrupts than the one currently being serviced are able to interrupt the current ISR.

3. Send an EOI to the PIC (send a 2ØH to the Mult I/O port BASE+4 of GROUP Ø). This will allow the current ISR to be interrupted by an interrupt of the same or lower priority.

4. Execute a RET instruction. Since the ISR was evoked through a CALL it must be exited with a RET or the stack pointer will not be positioned correctly.

Some Notes and Cautions

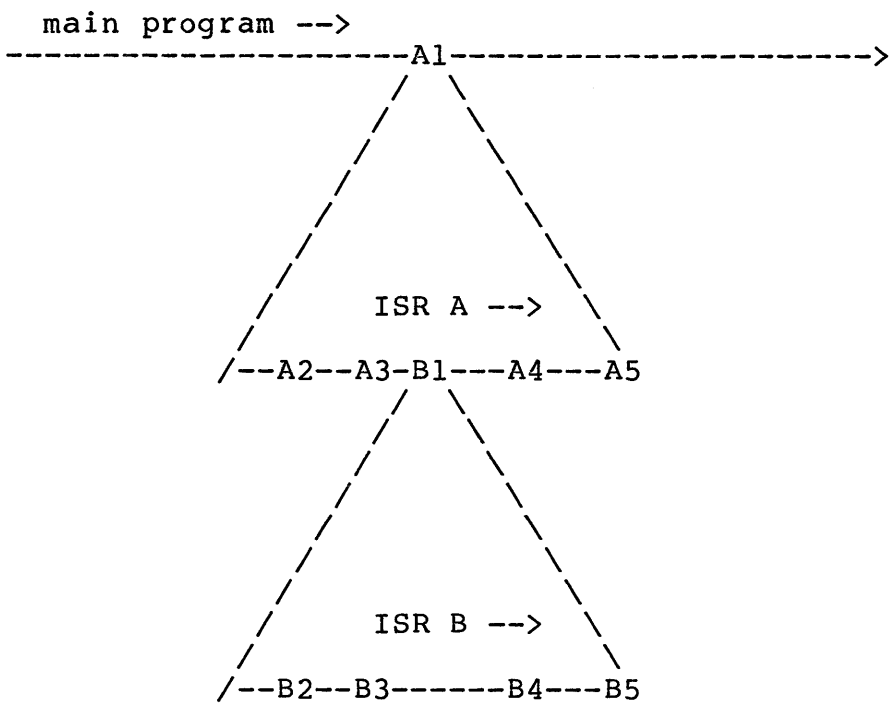
In situations where an ISR is to be allowed to be interrupted by another ISR, care should be taken to preserve CPU registers which might be altered so as to sabotage the interrupted service routine. By the same token, routines that are in any way time dependent should be written very carefully to preserve their integrity in case they are interrupted. For example, if two routines use the same ACE device, it is possible for a routine to check, say, the TBE status bit, find the device to be ready, prepare to send data to the device, get interrupted, and proceed, when control is regained, to send data to a device that may no longer be ready.

The CP/M * operating system contains a utility program, DDT, which is very useful in developing software. This program was intended by the writers to be used in an interrupt environment. When the 'G' or 'T' command is used, DDT temporarily disables interrupts (with a DI) in preparation for the run or trace, and then RE-ENABLES INTERRUPTS. This makes life difficult when one doesn't want interrupts enabled at all.

The following page gives a graphic illustration of the program flow which occurs when a program is interrupted and the ISR which results is itself interrupted.

* CP/M is a trademark of Digital Research

Illustration of Two Levels of Priority Interrupts



A1: Main program is interrupted by Interrupt Request A and PIC vectors program off to Interrupt Service Routine A (ISR A).

A2: ISR A removes the cause of its interrupt.

A3: ISR A issues an EI (Enable Interrupts) command to the CPU. This permits the servicing of a HIGHER priority interrupt.

B1: IRQ B (Interrupt Request B), a higher priority than IRQ A, causes ISR A to be interrupted, and the PIC vectors the program off to ISR B.

B2: ISR B removes the cause of its interrupt.

B3: ISR B issues an EI command to the CPU. ISR B may now in turn be interrupted by a higher priority IRQ.

B4: ISR B issues an EOI (End of Interrupt) command to the PIC. ISR B may be interrupted by SAME or LOWER priority IRQ.

B5: ISR B exits its service routine with a RET instruction. Control returns to ISR A.

A4: ISR A issues an EOI command to the PIC.

A5: ISR A exits its service routine with a RET instruction. Control returns to the main program.

SWITCH SETTINGS

SWITCH 2D: Extended addressing of Ram/Eprom
(left)

1 - A23
2 - A22
3 - A21
4 - A20
5 - A19
6 - A18
7 - A17
8 - A16

'OFF' = 1
'ON' = Ø
Example:
If base address of Ram/Eprom is ØF000H, all 'ON' except 4 (A20) will give Ram/Eprom the extended address 10F000H.

SWITCH 7B: (middle) I/O Port Address, Phantom response, Ram write enable.

1 - Phantom — 'ON' allows board to respond to phantom.

2- A7 \
3 - A6 \
4- A5 > I/O Port Address.
5 - A4 / 'OFF' = 1
6 - A3 / 'ON' = Ø

7 - RØW — Ram 'Ø' (5D) 'ON' enables write.
8 - R1W — Ram '1' (6D) 'ON' enables write.

SWITCH 10C: (right) Ram/Eprom Address, Extended address disable Bank preset, Power-on jump, Ram/Eprom Exchange.

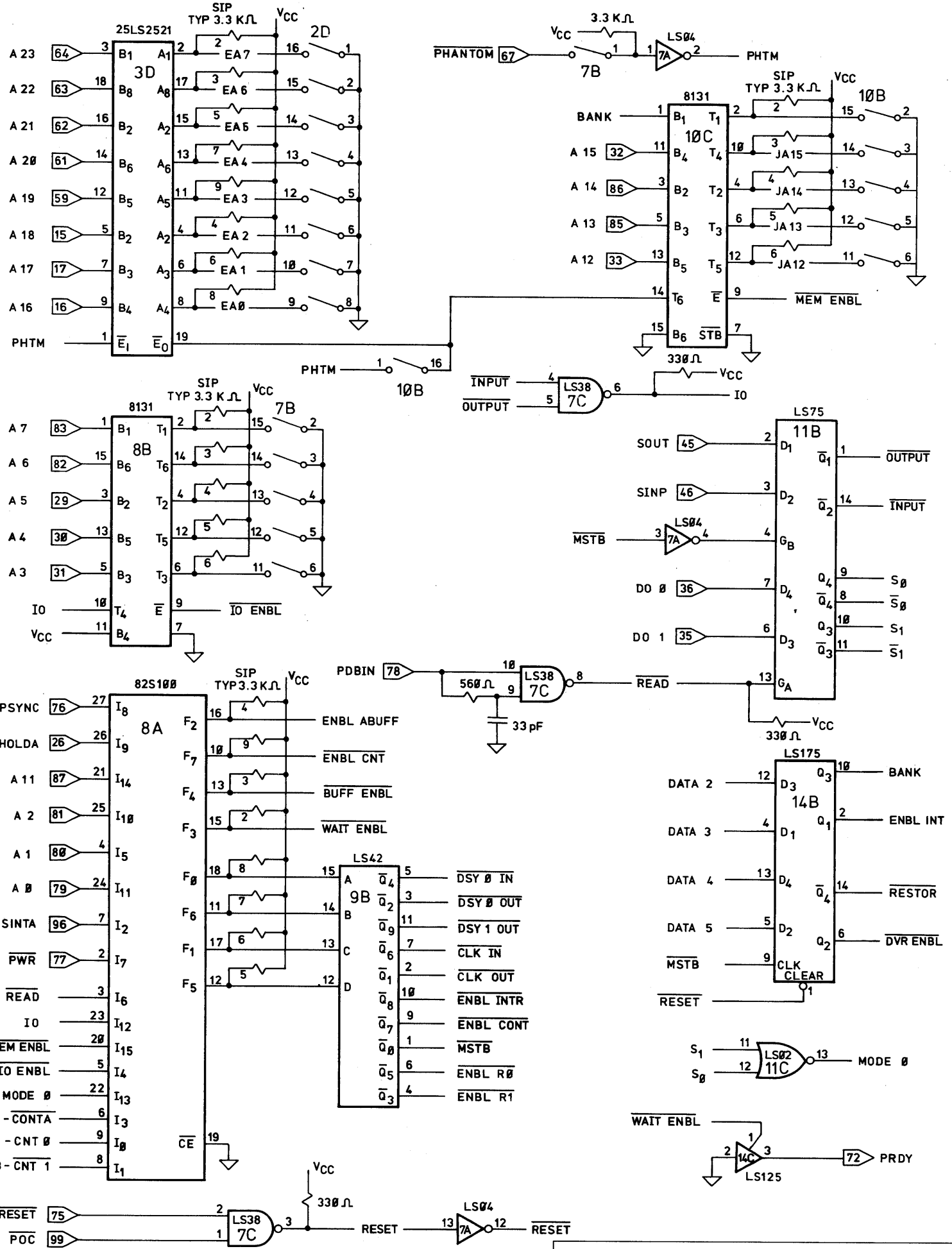
1 - Ext. addr. disable. — Note: Chip at location (3D) must be removed when 'ON'. 'ON' sets Ram/Eprom to respond regardless of extended address

2 - Ram/Epromenable. — 'ON' enablesRam/Epromon reset.

3 - A15 \
4 - A14 \ Ram/Eprom Address.
5 - A13 / 'OFF' = 1
6 - A12 / 'ON' = Ø

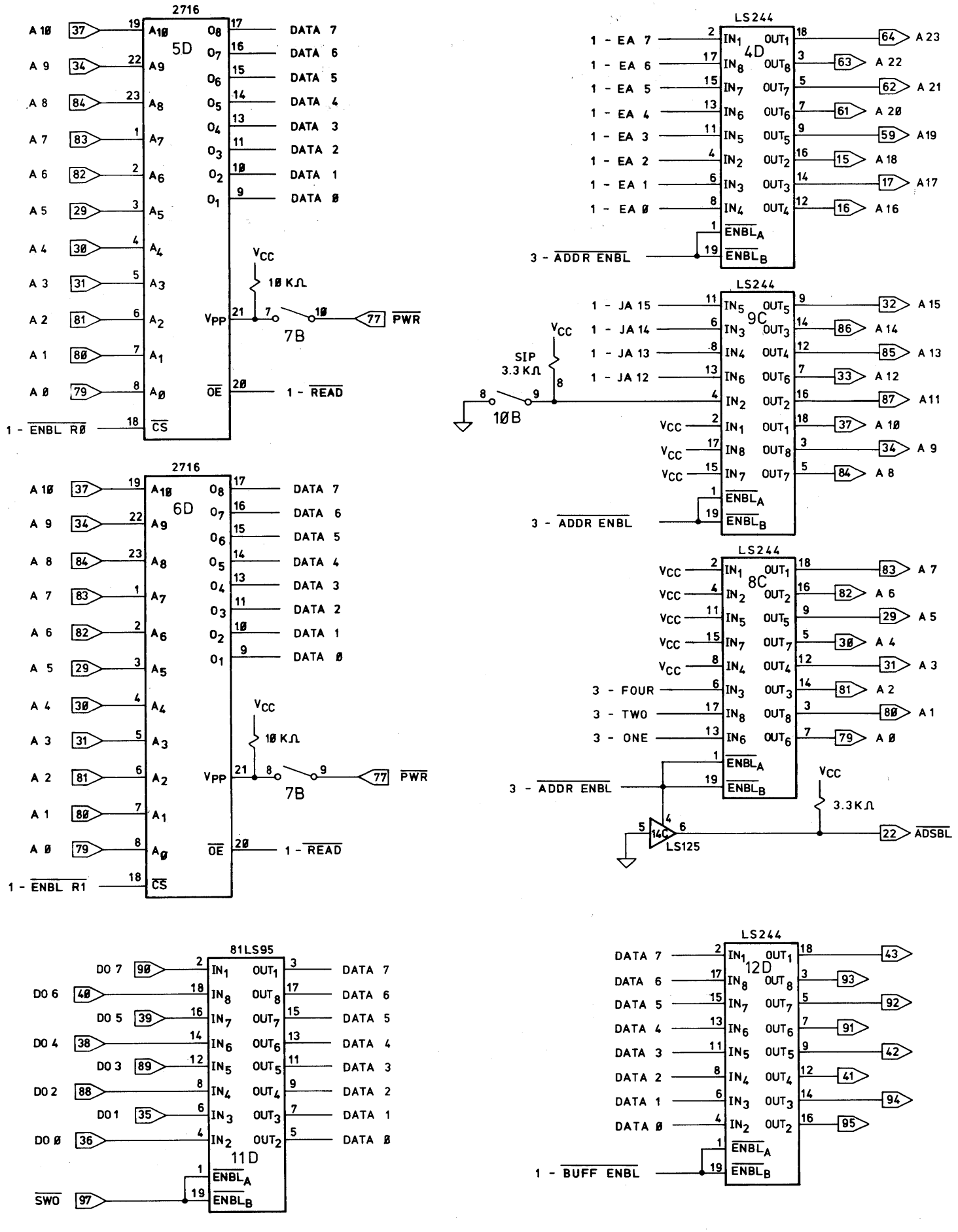
7- Power-onjump. — 'ON' enables POJ to Eprom.

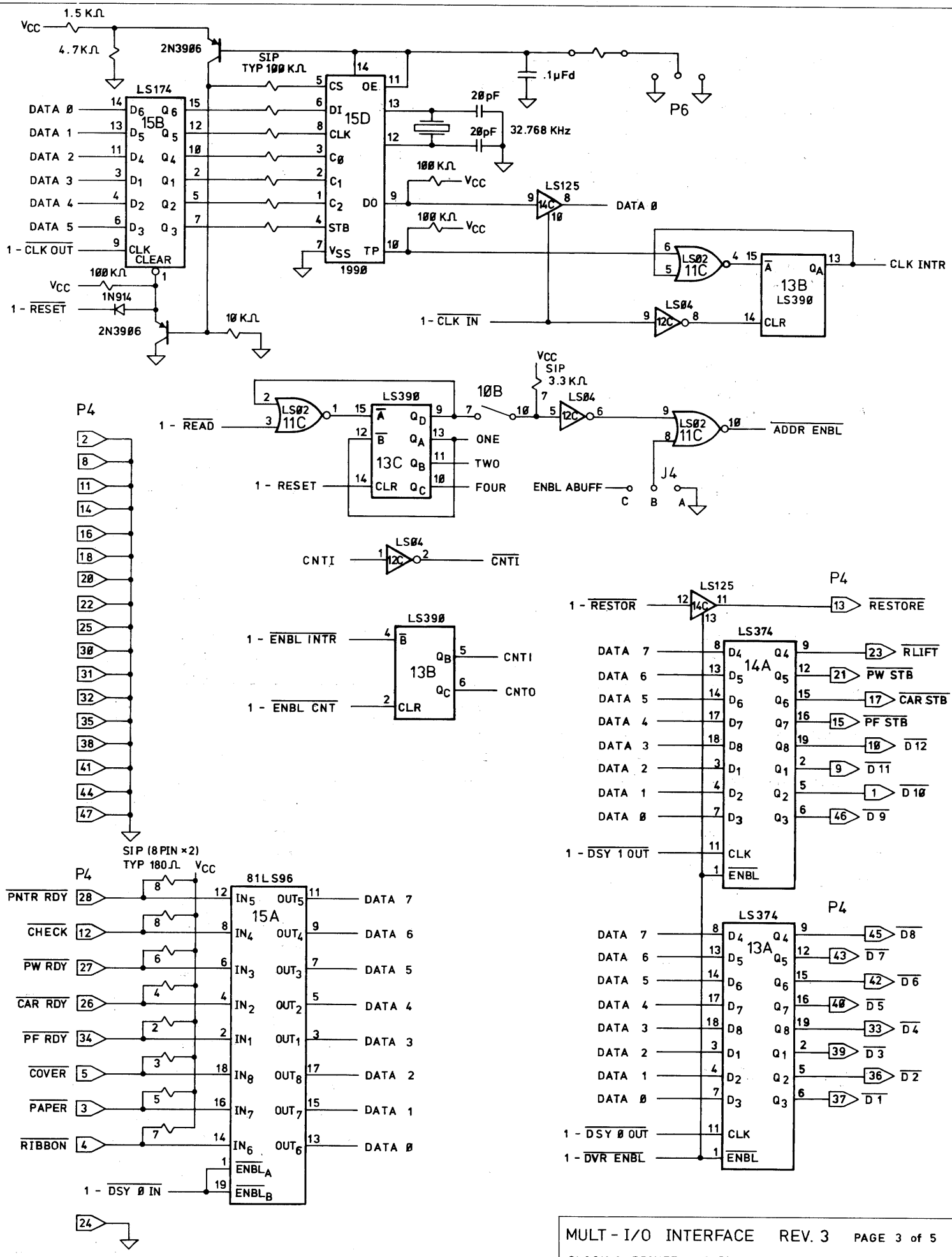
8 - POJ Eprom select. — 'ON' = RØ (5D), 'OFF' = R1 (6D).



MULT-I/O INTERFACE REV. 3 PAGE 1 of 5

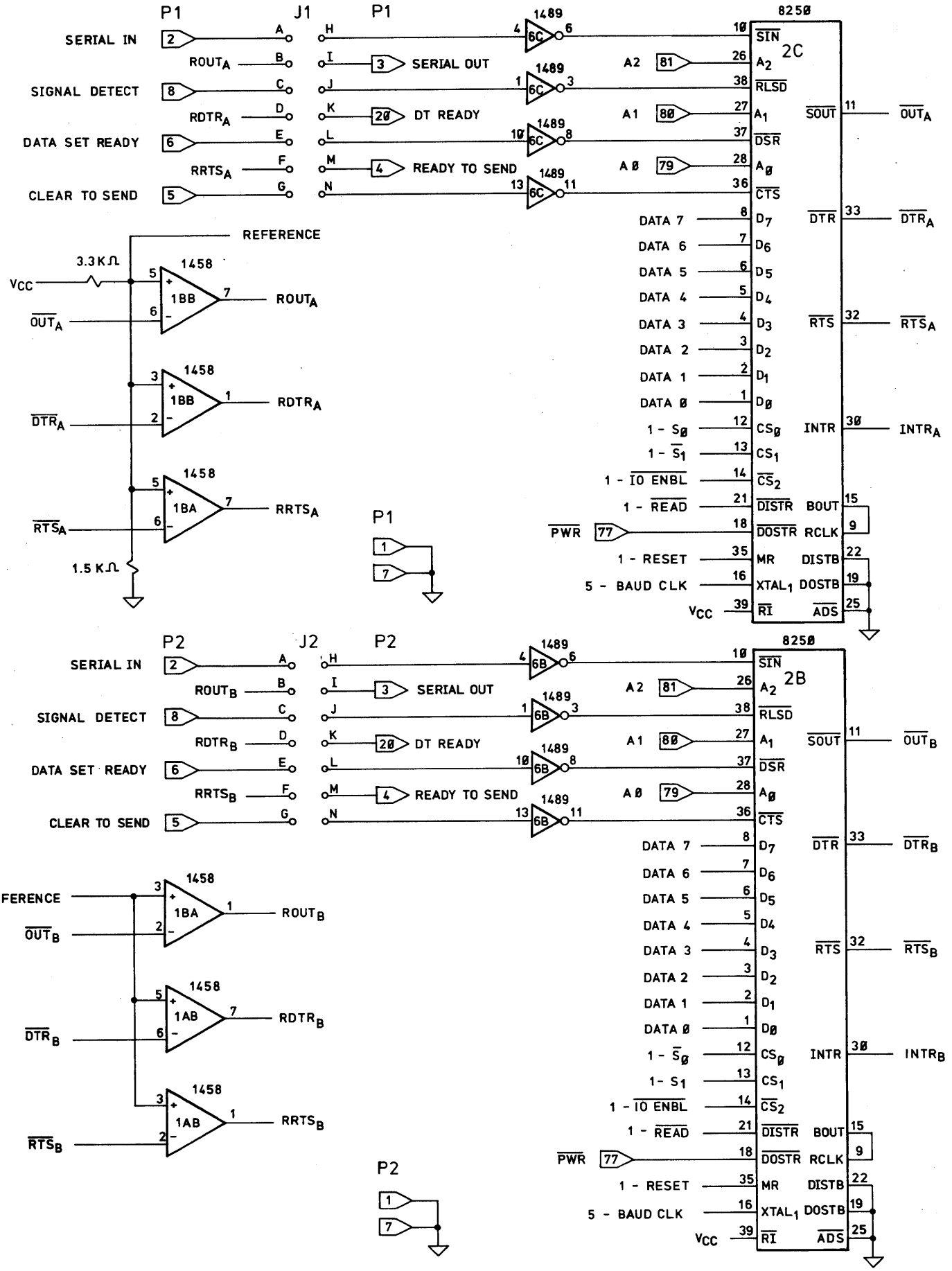
BOARD SELECT & DEVICE SELECT LOGIC ©1981 G. MORROW





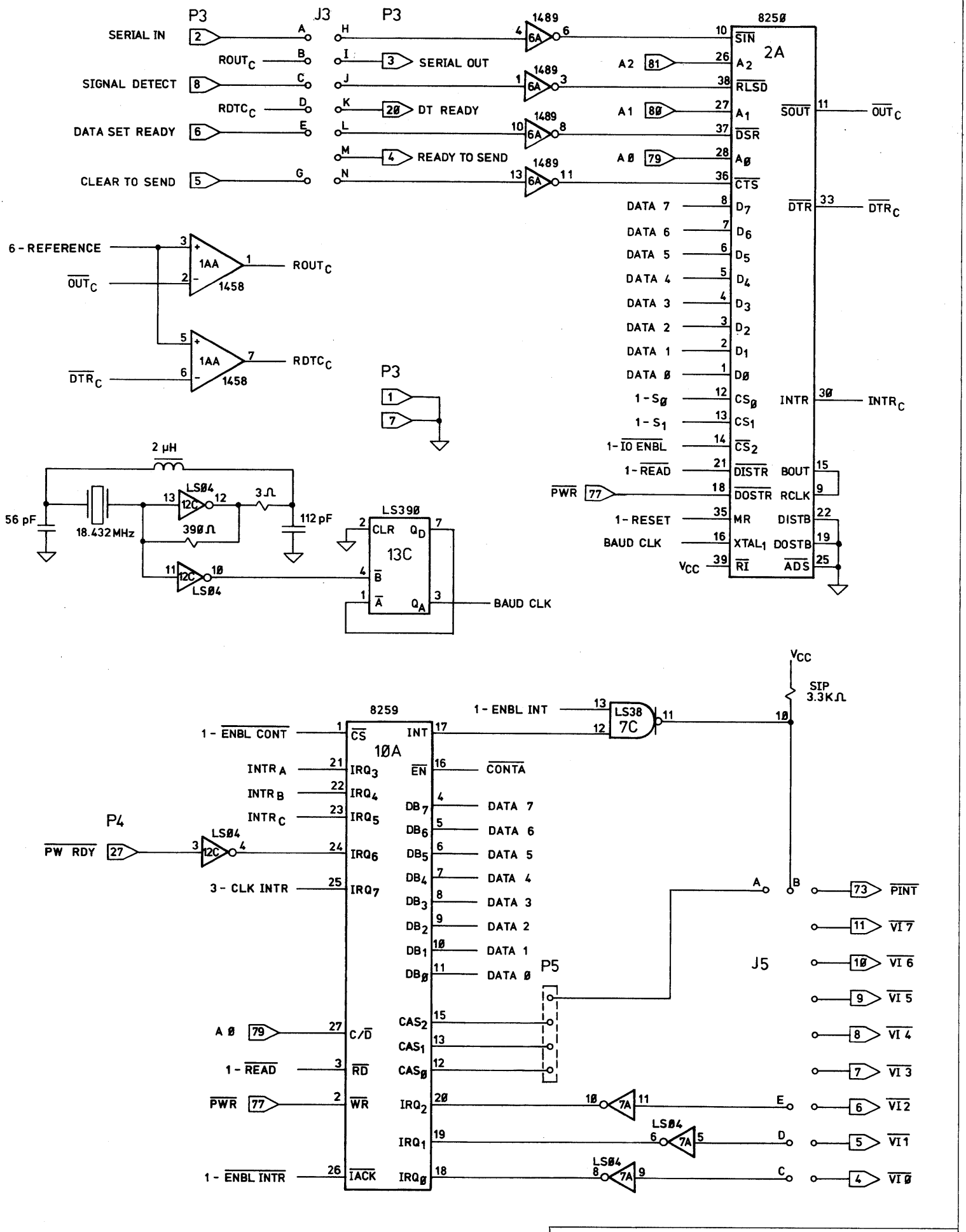
MULT - I/O INTERFACE REV. 3 PAGE 3 of 5

CLOCK & PRINTER LOGIC



MULT-I/O INTERFACE REV. 3 PAGE 4 of 5

SERIAL PORTS A & B



MULT-I/O INTERFACE REV. 3 PAGE 5 of 5

SERIAL PORT C & CONTROLLER LOGIC ©1981 G MORROW