THE ITHACA INTERSYSTEMS SERIES II MPU-80 A Z-80 PROCESSOR FOR THE IEEE 696 / S-100 BUS

InterSystems Publication 2000-1.O-U

ITHACA INTERSYSTEMS LIMITED WARRANTY

All equipment manufactured by ITHACA INTERSYSTEMS shall be guaranteed against defects in materials and workmanship for a period of ninety (90) days from date of delivery to the Buyer by the Seller, and the Seller agrees to repair or replace, at its sole option, any part which proves to be defective and attributable to any defect in materials or workmanship.

EXCEPT FOR THE WARRANTIES THAT THE GOODS ARE MADE IN A WORKMANLIKE MANNER AND IN ACCORDANCE WITH THE SPECIFICATIONS SUPPLIED, SELLER MAKES NO WARRANTY EXPRESS OR IMPLIED, AND ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE WHICH EXCEEDS THE FOREGOING WARRANTY IS HEREBY DISCLAIMED BY SELLER AND EXCLUDED FROM ANY AGREEMENT.

Buyer expressly waives its rights to any consequential damages, loss or expense arising in connection with the use of or the inability to use its goods for any purpose whatsoever.

No warranty shall be applicable to any damages arising out of any act of the Buyer, his employees, agents, patrons or other persons.

In the event that a unit proves to be defective, and after authorization by Seller, the defective part and/or unit, as authorized, must be securely packaged and returned Freight Prepaid by the Buyer to ITHACA INTERSYSTEMS for repair. Upon receipt of the unit, ITHACA INTERSYSTEMS will repair or replace, at its sole option, the defective part or product and return such part/product Freight Prepaid to the Buyer.

The remedies set forth herein are exclusive and the liability of Seller to any contract or sale or anything done in connection therewith, whether in contract, in tort, under any warranty, or otherwise, shall not, except as expressly provided herein, exceed the price of the equipment or part on which said liability is based.

This warranty is given solely to the original Buyer. No employee or representative of Seller is authorized to change this warranty in any way or grant any other guaranty or warranty.

InterSystems Series II

InterSystems Series II products present the first integrated computer system designed entirely to IEEE S-100 Standards. As such, it represents a powerful and versatile family of computers and accessories you can apply to virtually any computing task.

With Series II you have considerable flexibility in systems building, because the Series II architecture is modular and not tied to any particular type of computing. This modular and structural stability enables you to adapt your system easily to your changing computing needs.

As with all InterSystems Series II Products, the Series II Z-80 has been designed to take full advantage of the extensions and enhancements of the new IEEE S-100 bus specification, and yet remain compatible with the majority of pre-standard S-100 boards.

This owner's manual has been prepared to acquaint you with your processor and to serve as an aid in achieving its optimum use. Section 1 introduces the Series II Z-80 and provides some general information; Section 2 describes the use of the board in a system, how to use the memory management unit, vectored interrupt controller and other board features, and gives programming examples; Section 3 is a step-by-step guide to setup of the board for operation in your system, and the remaining sections provide technical details, parts list, and schematic diagram.

Your Series II processor has been fully tested and burned in at the InterSystems factory, and should work the first time in your system. If you have any problems with the unit in your system, give us a call at (607) 257-0190; our technical support personnel will be glad to assist you.

TABLE OF CONTENTS

Section 1

Introduction and General Information

1.1 Introducing the Series II Z-80

1.2 Service Information

Receiving Inspection
Replacement Parts
Factory Service
Contacting InterSystems

1.1 Introducing the InterSystems Series II Z-80

The InterSystems Series II Z-80 board is the first S-100 processor board to take full advantage of the recent IEEE specification. It has many special features that make it both powerful and easy to use. Among these features are:

1) S-100 bus cycle generator creates IEEE standard bus cycle timing for all bus operations to guarantee compatibility with all IEEE S-100 products.

2) A simple address management system provides two 4 kilobyte segments relocatable anywhere in the first megabyte of the 16 megabytes of the S-100 address space. Four light-emitting-diodes indicate the four most-significant address bits.

3) Vectored Interrupt Controller gives sophisticated, high speed interrupt handling with individual masking, fixed or rotating priorities, and optional polled operation.

4) 2 or 4 mHz processor speed is on-board pin-jumper selectable.

5) PROM monitor socket will accept 2708, 2716, 2758, or 2732 EPROM's and a reset jump is provided to the PROM. The PROM may be addressed at any 1 K boundary in the 64 K address space.

6) On-board wait generator optionally adds a single wait state to any Instruction Fetch, Memory Reference, Input/Output Reference, or on-board PROM Reference.

7) On-board circuit conducts the IEEE specified nested bus transfer for glitch-free DMA operations, and DMA controllers need not duplicate the circuit.

8) Operation without a front panel is supported both by optional MWRT generation, and by a specially designed front-panel connector which includes the Reset, Jump Enable, and GND signals from the processor card.

9) Special engineering features include 2 self-test modes for quick circuit debugging and Address and Status latching to reduce bus noise and improve reliability.

1.2 Service Information

Receiving Inspection

When your InterSystems Processor Module arrives, inspect both the equipment and the shipping carton immediately for evidence of damage during transit. If the shipping carton is damaged or water-stained, request the carrier's agent to be present when the carton is opened. If the carrier's agent is not present when the carton is opened, and the contents of the carton are damaged, save the carton and packing material for the agent's inspection. Shipping damages should be immediately reported to the carrier. Do not attempt to service the board yourself as this will void the warranty.

We advise that in any case you should save the shipping container for use in returning the module to InterSystems, should it become necessary to do.

Factory Service

InterSystems provides a factory repair service for all of its products. Before returning the module to InterSystems, first obtain a Return Authorization Number from our Sales Dept. This may be done by calling us, sending us a TWX, or by writing to us. After the return has been authorized, proceed as follows:

1) Write a letter describing the problem as best you can.

2) Describe your system to us, list boards by Manufacturer and name.

3) Include Xerox copies of the schematics of boards by manufacturers other than InterSystems.

4) Include the Return Authorization Number.

5) Pack the above information in a container suitable to the method of shipment.

6) Ship prepaid to InterSystems.

Your module will be repaired as soon as possible after receipt and return shipped to you prepaid.

Contacting InterSystems:

The following apply both for correspondence and service.

Ithaca InterSystems Inc.
1650 Hanshaw Rd.
P.O. Box 91
Ithaca N.Y. U.S.A.
14850

Telephone (607) 257-0190
TWX 510 255-4346

In Europe:

Ithaca InterSystems (U.K.) Ltd.
58 Crouch Hall Rd
London N8 8HG. U.K.

Telephone 01-341-2447
Telex 299568

Section 2.0

Series II Z-80 Architecture

2.1 System Design Overview

2.2 Address Management for Large Systems
    Memory Management Subroutines

2.3 Interrupt Handling with the Series II Z-80

2.4 Z-80 Processor Interrupt Modes
    Mode 0
    Mode 1
    Mode 2

2.5 Interrupt Controller Architecture

2.6 Interrupt Controller Operating Options
    Fixed or Rotating Priorities
    Common and Individual Vectoring
    Polled Mode
    Master Mask
    Status Register
    Interrupt Service Register
    Information Transfers

2.7 Commands to the Interrupt Controller

2.8 Using the Interrupt Controller
    Vectored Interrupt Controller Handlers

2.9 Series II Bus Interface

2.10 Direct Memory Access Operations

2.0 Series II Z-80 Architecture

The Series II architecture supports a full range of system requirements, from small, dedicated applications to complex, multi-tasking systems. As a modular computing system, it is a flexible base from which to build a system tailored to your computing needs and achieve:

* Outstanding price/performance for all size systems.
* High standards of reliability and serviceability through the use of proven technologies, close adherence to the IEEE S-100 standard, and modular construction.
* Expandability to meet your future needs.

2.1 System Design Overview

InterSystems Series II offers an integrated hardware and software system developed for the IEEE S-100 bus. At the core of the system are two compatible processors, the Series II Z-80 and the Series II Z-8000, offering the most sophisticated processing available for both 8 bit and 16 bit computer systems.

Memory requirements for both 8 and 16 bit compatible systems are handled by two new Series II memory boards, a 16 K static memory board and a 64 K dynamic memory board. Both these unique memories may be used in either 8 or 16 bit IEEE systems without modification and without wait states; the memory data bus automatically adjusts itself to the word width requested by the processor.

Series II Input/Output boards complete the hardware side of the system. Our multiple I/O board provides extremely versatile connection to all kinds of peripheral devices. The board provides 2 full RS-232 serial ports, 4 parallel ports, plus 8 bit addressable I/O lines for event sensing and control, and on-board interrupt prioritization.

Other Series II I/O boards include a high speed DMA floppy disk controller, capable of both single and double density; and an 8 bit, 8 channel Analog Input/Output board, as well as a forthcoming 12 bit analog I/O system. These sophisticated Input/Output systems are designed to contribute to the total efficiency of the Series II system, offering a flexible, modular approach for building your computing facility to meet your needs now and well into the future.

Software support for the Series II system revolves around our unique compiler, PASCAL/Z, available now for our 8-bit Z-80 system and available

soon for the 16 bit Z-8000 system. PASCAL/Z is rapidly becoming the industry standard for high speed, high efficiency Pascal programming.

Unlike most single card computer boards -- typically optimized for dedicated process control rather than for system support -- the Series II Z-80 offers a set of features that provide the core of a powerful general-purpose computer system. Instead of on-board random access memory, the Series II Z-80 provides segmented memory management. Instead of on-board Input/Output, the Series II provides Vectored Interrupt Control for the most efficient use of all system Input/Output. Instead of stripping down the S-100 bus interface to use the fewest parts, the Series II Z-80 cards are designed to produce the most reliable and consistent bus cycles.

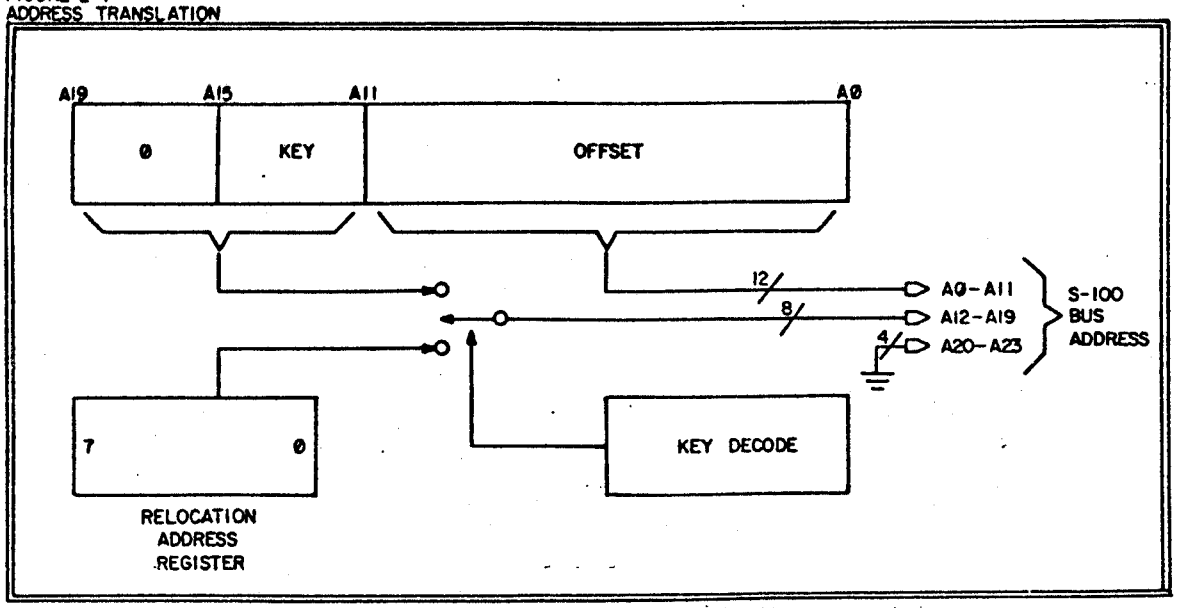
2.2 Address Management for Large Systems

A dual segment address translator permits the Series II Z-80 to address memory beyond the 64 Kilobytes directly addressed by the processor chip itself. In this mode the Series II Z-80 card generates addresses that are 20 bits long, allowing access to any location in the first megabyte of the 16 megabyte S-100 address field. The extended address bits generated by the Series II Z-80, A16 throuh A19, are visually indicated by the four LED's in the upper left corner of the board. These lights indicate the complement of the address asserted on the bus, that is, a zero on the bus is indicated by a lighted LED. (The Series II Z-80 card always asserts the highest four address bits of the IEEE 24 bit extended address range -- A20 through A23 -- as zeros).

The address translator is configured as an 8 K "window" in the normal 64 K address space; the location of the window is set on the board with jumpers. When this window is addressed by the processor, the most significant four bits of the processor's address are replaced with a new four bits; in addition, four extended bits are also asserted. The remaining low address bits -- A0 through A11 -- pass through the translator unchanged.

This new, extended address can access memory throughout a 1 Megabyte address space. The value of the "replacement" address -- the 8 most significant bits A12 to A19 -- is set dynamically on the card, by storing data in registers provided for this purpose. There are two such "relocation" registers, and the 8 K "window" is in fact split into two 4 K sections, so that two entirely different areas of the 1 Megabyte address space can be accessed conveniently. The address translation process is shown in Figure 2-1.

FIGURE 2-1
ADDRESS TRANSLATION

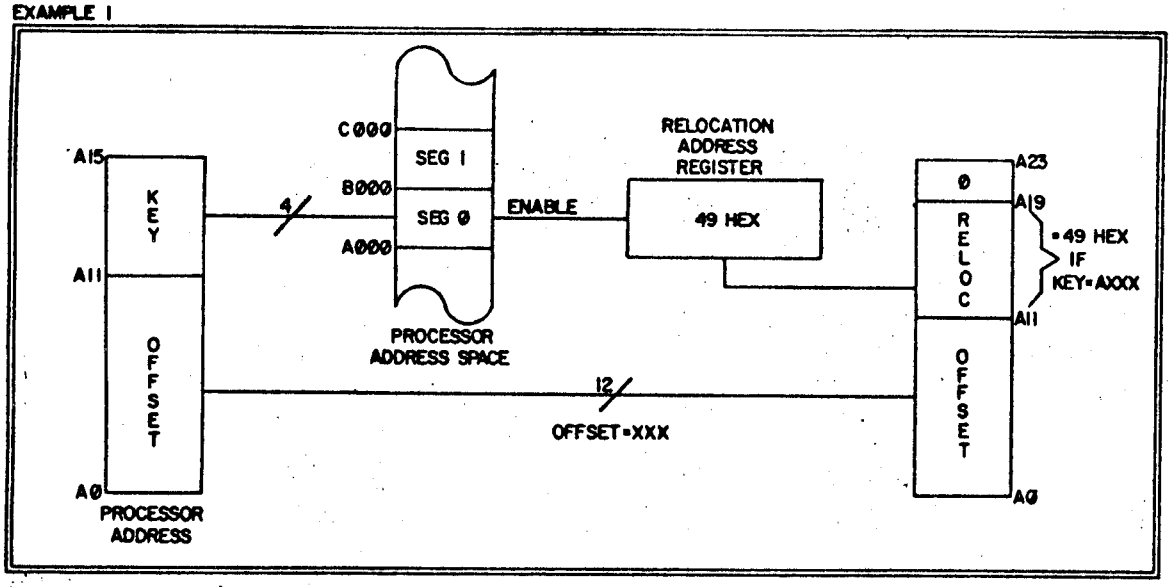


Note that the area of main memory designated as the "window" is not lost to the processor: the relocation registers may be loaded such that the translated address is equal to the address generated by the processor before translation, thus providing "transparent" address mapping.

Some programming examples will help to illustrate the operation of the address translator. While the "window" area may be hardware addressed to any 8 K boundary, for the purpose of these examples we will assume that the 8 K window occupies the addresses from the 41st K to the 48th K (hex A000 to BFFF). This means that the first segment, SEG 0, is addressed from address A000 to AFFF, and the second segment, SEG 1, is addressed from B000 to BFFF (the card is shipped with the window addressed in this area).

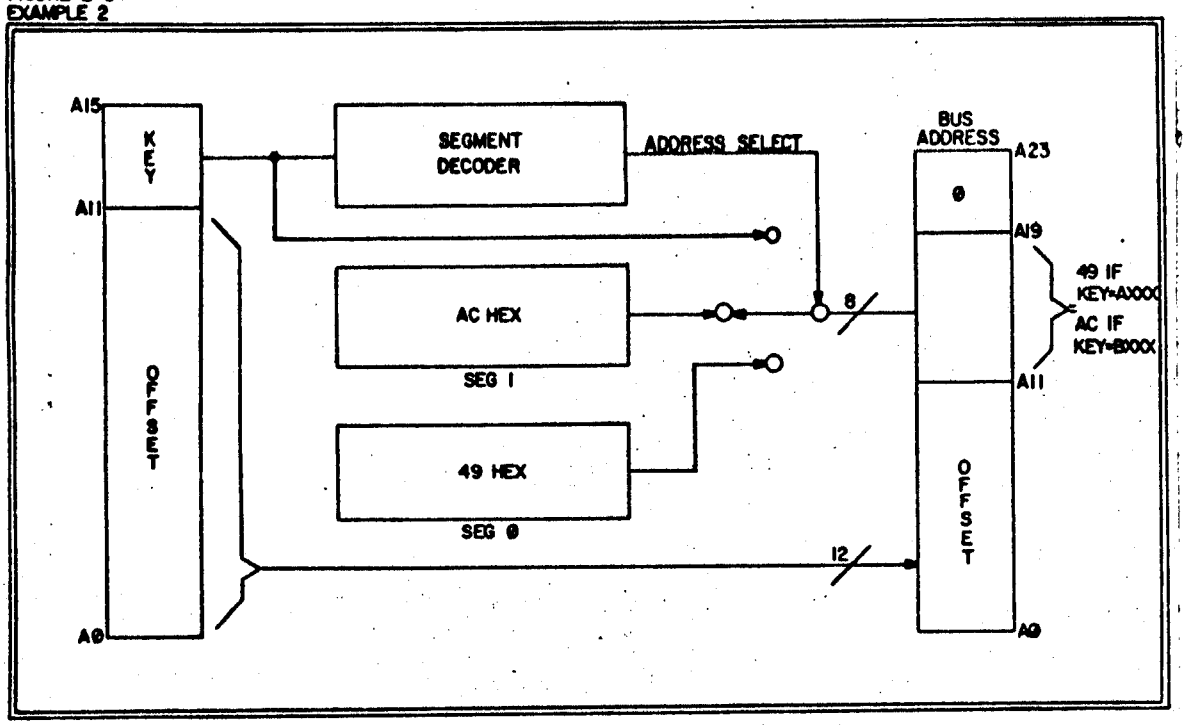
As a first example, suppose we wish to move the first 2 K bytes of main storage (address 00,0000 to 00,07FF) to a block beginning in the extended storage area, say at address 04,9000 (hex). This may be easily done by loading the SEG 0 relocation register with the address 49 (hex) and transfering the data from locations 0 to 7FF to locations A000 to A7FF. The processor address "Axxx" ("x" is used to denote an arbitrary hex digit) selects the segment 0 translation. The top four bits of the processor's address ("A") are replaced by the contents of the relocation register, in this case 49 hex, and the low 12 bits of the processor's address are passed to the bus unchanged: The resulting bus address is 04,9xxx hex. See Figure 2-2.

FIGURE 2-2
EXAMPLE 1



As a second example, suppose we wish to move a 1 K block of data between two separate areas in the extended address space, say from 04,9000 to 0A,C000. If we load the SEG 0 relocation register with 49 and the SEG 1 register with AC, the transfer is accomplished by moving the contents of locations A000 thru A3FF to B000 thru B3FF. In this case the segment 0 designator (address Axxx) is replaced by address 49xxx, and the segment 1 designator (address Bxxx) is replaced by address ACxxx. (Figure 2-3)

FIGURE 2-3
EXAMPLE 2



Programs may also be run in the extended storage area. In this kind of operation the two segments may be grouped consecutively, giving a single 8K segment for the program. Programs should be assembled to run at address A000 and should be less than 8K bytes in length unless checking is done to assure that the program counter won't overflow back into main storage.

Even with these restrictions, this technique is extremely powerful for managing a set of overlays for a sophisticated program. All overlay programs may be assembled to run at address A000, but any number of overlays may be stored at random in the extended address space. Any of the stored overlays may be activated by loading the relocation registers with the appropriate pointers, and calling the overlay at address A000. Note that any location in the main storage area may be directly referenced by the current overlay, but that calls to other overlays should be routed through a kernel which will switch the requested overlay into the map area, and switch the first overlay back in on the return from the requested function.

The following programs are a simple set of subroutines which open either of the segments onto the specified areas of the system memory. Note that the routines do not limit the mappings to the extended storage area; any location in the first megabyte of memory may be accessed by these routines. For the purposes of simplicity, these subroutines will always consider the zero segment as an input file and the one segment as an output file, though

there is no physical reason why both the segments may not simultaneously be input and output files.

The routines below initialize the memory management unit; open the input and the output files, get the next byte from the input file; put the next byte to the output file; check for window overflow and correct the relocation pointers if overflow occurs; and will move up to 64 K bytes from any location in memory to any other.

These simple routines could easily be combined with others to provide multiple open files, searching, sorting, overlay management, dynamic memory allocation, and a variety of other sophisticated system functions.

```
;            MEMORY MANAGEMENT SUBROUTINES
;
;DEFINITIONS
IOBASE  EQU     0E0H    ;BASE ADDRESS OF ON-BOARD I/O
REG0    EQU     IOBASE+2;ADDRESS OF RELOCATION REGISTER 0
REG1    EQU     IOBASE+3;ADDRESS OF RELOCATION REGISTER 1
BASE    EQU     0AH     ;RELOCATION POINTER FOR INITIALIZATION
KEY0    EQU     0A0H    ;SEGMENT 0 ADDRESSED AT A000 HEX
KEY1    EQU     0B0H    ;SEGMENT 1 ADDRESSED AT B000 HEX
;
;ENTRY POINTS AND VARIABLES
;
INIT:   JMP     INIT0
OPNIN:  JMP     OPNIN0
OPNOT:  JMP     OPNOT0
GETB:   JMP     GETB0
PUTB:   JMP     PUTB0
MOVBLK: JMP     MOVBK0
;
;INPUT VARIABLES
INBASE  DS      3
OTBASE  DS      3
LEN     DS      2
;
;LOCAL VARIABLES
SEG0    DS      1
SEG1    DS      1
INPTR   DS      2
OTPTR   DS      2
SEG     DS      1
;
;SUBROUTINE INIT: INITIALIZE THE MEMORY MANAGEMENT UNIT
;SUCH THAT THE SEGMENTS POINT TO THE SAME AREAS THEY PHYSICALLY
;OCCUPY IN THE MAIN MEMORY. IN THIS WAY THE RELOCATION DEVICE
;IS LOGIALLY TRANSPARENT TO THE SYSTEM.
;USE A
;
INIT0:  MVI     A,BASE  ;LOAD A WITH BASE ADDRESS OF SEGMENT 0
        OUT     REG0    ;SETUP SEGMENT 0
        INR     A       ;A=BASE ADDRESS OF SEGMENT 1
        OUT     REG1    ;SETUP SEGMENT 1
        RET             ;EXIT
;
```

```
;SUBROUTINE OPNIN:  OPEN AN INPUT FILE IN THE EXTENDED ADDRESS
;AREA, BEGINNING AT THE ADDRESS GIVEN IN THE MEMORY LOCATIONS
;CALLED INBASE.  SET THE SEGMENT 0 REGISTER WITH THE APPROPRIATE
;OFFSET AND SAVE AT SEGO.  INPNTR IS THE POINTER TO THE FIRST
;BYTE OF THE INPUT FILE.
;USE A,F,D,E,H,L
;FORMAT OF INBASE IS INBASE=LEAST SINIFICANT BYTE, INBASE+1=
;MID-SIGNIFICANT BYTE, INBASE+3=MOST SIGNIFICANT BYTE (NOTE
;THAT BITS 4 THRU 7 OF THE MSB MUST BE 0'S)
;
;
OPNINO: LDED    INBASE  ;LOAD DE WITH TWO LOW ORDER BYTES OF
                        ;INBASE
        LDA     INBASE+2;A=MSB OF INBASE
        CALL    SPNT    ;CALCULATE SEG AND OFFSET FROM A,D,E
        STA     SEGO    ;SAVE SEGMENT POINTER AT SEGO
        OUT     REGO    ;SETUP THE SEGMENT REGISTER
        MOV     A,H
        ORI     KEYO    ;INPNTR = A000+OFFSET
        MOV     H,A
        SHLD    INPTR   ;SAVE AT INPNTR
        RET             ;EXIT
;
;
;SUBROUTINE OPNOT: OPEN AN OUTPUT FILE BEGINNING AT THE ADDRESS
;CONTAINED AT OTBASE.  SET THE SEGMENT 1 REGISTER WITH THE
;RELOCATION ADDRESS AND SAVE AT LOCATION SEG1.  THE POINTER
;AT OTPTR IS THE OFFSET THAT POINTS TO THE FIRST BYTE OF THE
;OUTPUT FILE.
;USE A,F,D,E,H,L
;
;
OPNOTO: LDED    OTBASE
        LDA     OTBASE+2;LOAD A,D,E WITH ADDRESS IN OTBASE
        CALL    SPNT    ;CALCULATE RELOCATION AND OFFSET
        STA     SEG1    ;SAVE RELOCATION POINTER
        OUT     REG1    ;LOAD RELOCATION REG FOR SEGMENT 1
        MOV     A,H
        ORI     KEY1    ;OFFSET+B000 KEYS FOR SEGMENT 1
        MOV     H,A
        SHLD    OTPTR   ;SAVE AT OTPTR
        RET             ;EXIT
;
```

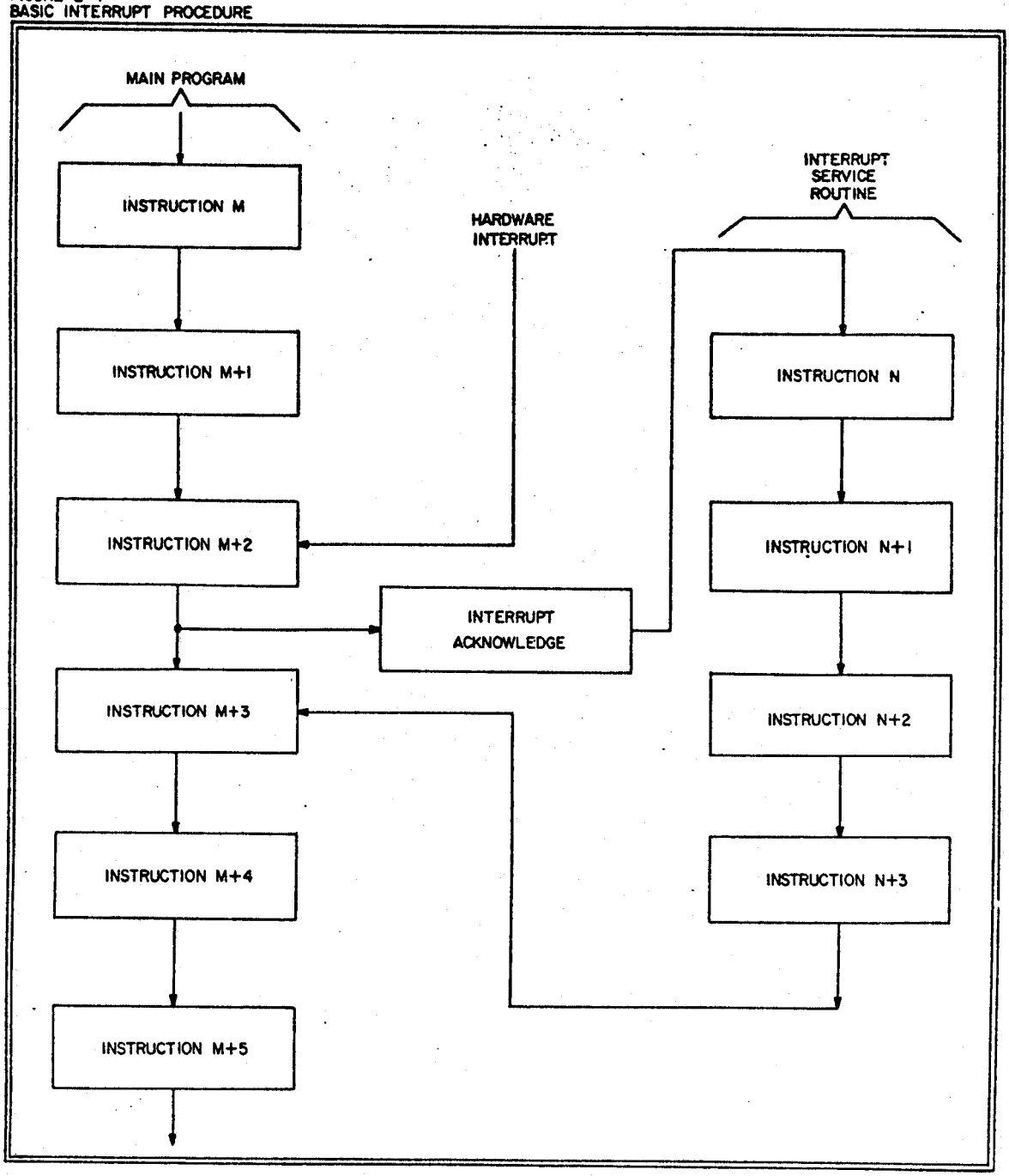
```
;SUBROUTINE SPNT:  CALCULATE THE RELOCATION POINTER AND THE
;OFFSET FOR LOADING TO THE MEMORY MANAGEMENT UNIT FROM THE
;20 BIT ADDRESS CONTAINED IN A,D,E (MSB TO LSB).
;RETURN THE RELOCATION POINTER IN A AND THE OFFSET IN HL.
;USE A,F,D,E,H,L
;
SPNT:   LXI     H,SEG   ;HL POINT TO SEG
        MOV     M,D     ;SEG GETS MID-BYTE OF BASE ADDRESS
        RRD             ;ROTATE DIGITS USING A AND SEG
        ANI     OFH     ;MASK BITS 4-7 OF OFFSET
        MOV     H,A
        MOV     L,E     ;HL GET OFFSET
        LDA     SEG     ;A GETS RELOCATION POINTER
        RET             ;EXIT
;
;SUBROUTINE GETB: GET THE NEXT BYTE FROM THE INPUT FILE,
;AND INCREMENT THE INPUT POINTER, INPTR.  CHECK FOR WINDOW
;OVERFLOW AND RECALCULATE THE OFFSET AND SEGMENT RELOCATION
;ADDRESS ON OVERFLOW.
;USE A,F,HL,DE
;RETURN THE BYTE IN A.
;
GETBO:  LHLD    INPTR   ;HL POINT TO NEXT BYTE
        MOV     D,M     ;BYTE IN D
        INX     H       ;INPNTR=INPNTR+1
        MOV     A,H
        CPI     KEYO+10H;CHECK FOR SEGMENT OVERFLOW
        JNZ     EXGET   ;NO OVRFLW
        MVI     H,KEYO  ;OVRFLW, RESET INPNTR
        LDA     SEGO
        INR     A       ;SEGO=SEGO+1
        STA     SEGO    ;SAVE AT SEGO
        OUT     REGO    ;LOAD RELOCATION REGISTER WITH NEW VALUE
EXGET:  SHLD    INPTR   ;SAVE POINTER TO NEXT BYTE
        MOV     A,D     ;RESULT IN A
        RET             ;EXIT
;
```

```
;SUBROUTINE PUTB: PUT THE BYTE IN A INTO THE OUTPUT FILE.
;INCREMENT THE OUTPUT FILE POINTER, OTPTR, AND CHECK FOR
;OVERFLOW. IF THE POINTER IS OUT OF THE 4K WINDOW, RECALCULATE
;THE RELOCATION POINTER AND THE OFFSET.
;USE A,H,L
;
PUTBO:  LHLD    OTPTR   ;HL POINT TO INSERT POSITION
        MOV     M,A     ;STORE THE BYTE
        INX     H       ;OTPTR=OTPTR+1
        MOV     A,H
        CPI     KEY1+10H        ;OVERFLOW?
        JNZ     EXPUT   ;NO, GO ON
        MVI     H,KEY1  ;RESET OFFSET ON OVRFLW
        LDA     SEG1
        INR     A       ;SEG1=SEG1+1 ON OVRFLW
        STA     SEG1    ;SAVE NEW RELOCATION POINTER
        OUT     REG1    ;LOAD THE RELOCATION REGISTER

EXPUT:  SHLD    OTPTR   ;SAVE NEW OFFSET
        RET             ;EXIT
;
;SUBROUTINE MOVBLK: MOVE A BLOCK OF DATA OF THE LENGTH GIVEN IN
;LEN FROM THE ADDRESS GIVEN IN INBASE TO THE ADDRESS GIVEN IN
;OTBASE.  UP TO 64K BYTES MAY BE MOVED.
;USE A,F,B,C,D,E,H,L
;
MOVBKO: CALL    OPNIN   ;OPEN THE INPUT FILE AT INBASE
        CALL    OPNOT   ;OPEN THE OUTPUT FILE AT OTBASE
        LBCD    LEN     ;BC GET LENGTH COUNTER

MLP:    CALL    GETB    ;GET THE NEXT BYTE
        CALL    PUTB    ;STORE THE BYTE IN THE OUTPUT FILE
        DCX     B       ;LEN=LEN-1
        MOV     A,C
        ORA     B       ;BC=0?
        JNZ     MLP     ;NO, GO ON
        RET             ;YES, EXIT
;
```

2.3 Interrupt Handling with the Series II Z-80

All processor systems must transfer information to and from the user of the system, as well as process that information while it is in the system. Often computer systems have a number of peripheral devices connected to them, each with different requirements with regard to the frequency, urgency, and complexity of service needed from the system processor.

FIGURE 2-4
BASIC INTERRUPT PROCEDURE



There are two general methods for coordinating these service requirements of assorted peripheral devices in a computer system: program controlled service routines, and interrupt driven service.

In a program controlled service system, the processor is responsible for the initiation and control of all peripheral device transfers, while in an interrupt driven system the peripherals themselves signal the processor that they require service. This signal, called an interrupt, modifies the sequence of instructions that the processor executes to include a service routine for the signaling peripheral.

As the number of system peripherals grows, or as the complexity of the service they require of the system increases, software polling techniques for coordinating device service become increasingly inefficient; the polling program becomes very time consuming, and quickly consumes a significant fraction of the processing resource. Even more disastrous to the overall efficiency of the system is that it becomes impossible to guarantee that a particular device will be serviced within a given amount of time, unless the attention of the processor is totally devoted to such a time critical-event.

Interrupt driven systems, on the other hand, enhance the total throughput of the system by eliminating the need for software polling procedures, and have the additional advantage of priority resolution among simultaneous requests for processor service. A basic interrupt procedure is shown in Figure 2-4 (above). The instruction sequence of the main program is altered by the occurrence of a hardware interrupt at instruction M+2. The processor acknowledges the interrupt after the completion of the instruction, and branches to the interrupt service routine. When the interrupt service has been completed, the processor is free to return to the main program and continue execution.

The Series II Z-80 has been designed to support interrupt driven systems by the inclusion of a sophisticated vectored interrupt controller. The controller manages the masking of individual interrupts under software control, priority resolution among simultaneous service requests, and the vectoring for the 8 interrupt levels of the S-100 bus.

Additional features of the interrupt controller provide either a fixed priority mode for the resolution of multiple service requests, or a rotating mode such that all devices have, over time, equal priorities. The direct vectoring capability of the interrupt controller may also be bypassed and a polled mode option invoked under software control.

The integral mask register allows individual interrupts to be disabled or enabled by the processor. The mask register may be loaded either in parallel by the processor, or individual bits in the mask register may be controlled. The bus interrupt inputs to the controller use pulse-catching

circuits to sense interrupt requests, so that both one-shot and stable requests will be accepted by the system. Narrow noise pulses, however, are ignored.

With this combination of features, interrupt systems of any complexity may easily be implemented. The following sections describe the internal architecture of the interrupt controller, its command structure, and provide a sample interrupt service handler.

2.4 Z-80 Processor Interrupt Modes

The Z-80 processor chip has three basic modes of interrupt operation, which may be changed under software control. The Series II Interrupt Controller may be programmed to operate in any one of these modes; Mode 2, however, is by far the most powerful and versatile, and our programming examples will be confined to it.

Mode 0

Mode 0 is identical to the interrupt response mode of the 8080 processor chip. In this mode the interrupting device places an instruction on the processor's data bus during the Interrupt Acknowledge cycle. The processor then executes this instruction instead of the next instruction in memory. In theory any instruction may be placed on the data bus, but in practice only single byte instructions should be used because the Z-80 only produces an interrupt acknowledge cycle on the first byte of a multiple byte instruction. The single byte call instructions, called Restarts, are the instructions most often used in interrupt mode 0. These instructions execute a call to one of eight fixed locations in low memory, depending on the particular coding of the instruction. The interrupt controller may be programmed to supply any of the Restart instructions in response to any interrupt.

Mode 1

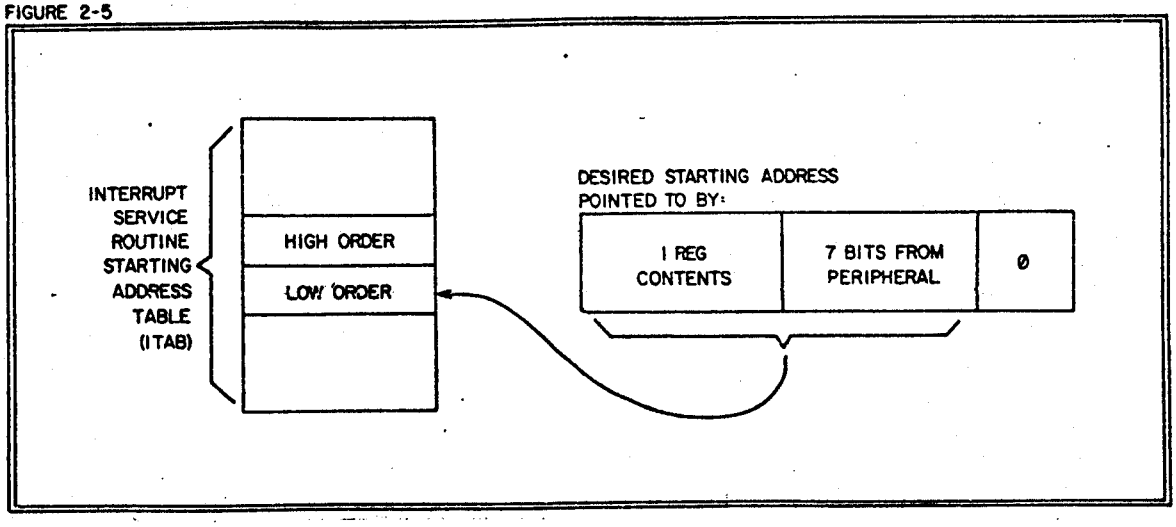
When interrupt mode 1 has been selected by the programmer, all interrupts in the system will be responded to by a call to location 066 Hex. A common service routine and dispatch table should begin at that location if mode 1 is selected.

Mode 2

Mode 2 is the most powerful interrupt response mode: a single byte response from the interrupt controller and an indirect call is made to any location in main memory.

In mode 2, the programmer maintains a table of 16-bit starting addresses for all of the system's interrupt service routines. The table, called ITAB in the programming examples, may be located anywhere in memory, as long as it does not cross a 256 byte boundary. When an interrupt is accepted, a 16 bit pointer to the table is formed by the processor from the contents of a special register, the I register, which supplies the most significant 8 bits of the pointer, and the vector supplied by the interrupt controller during the acknowledge cycle, which supplies the least significant 8 bits of the pointer. The Z-80 then fetches the service routine starting address from the table entry selected by the pointer, and performs a call to the service routine.

FIGURE 2-5



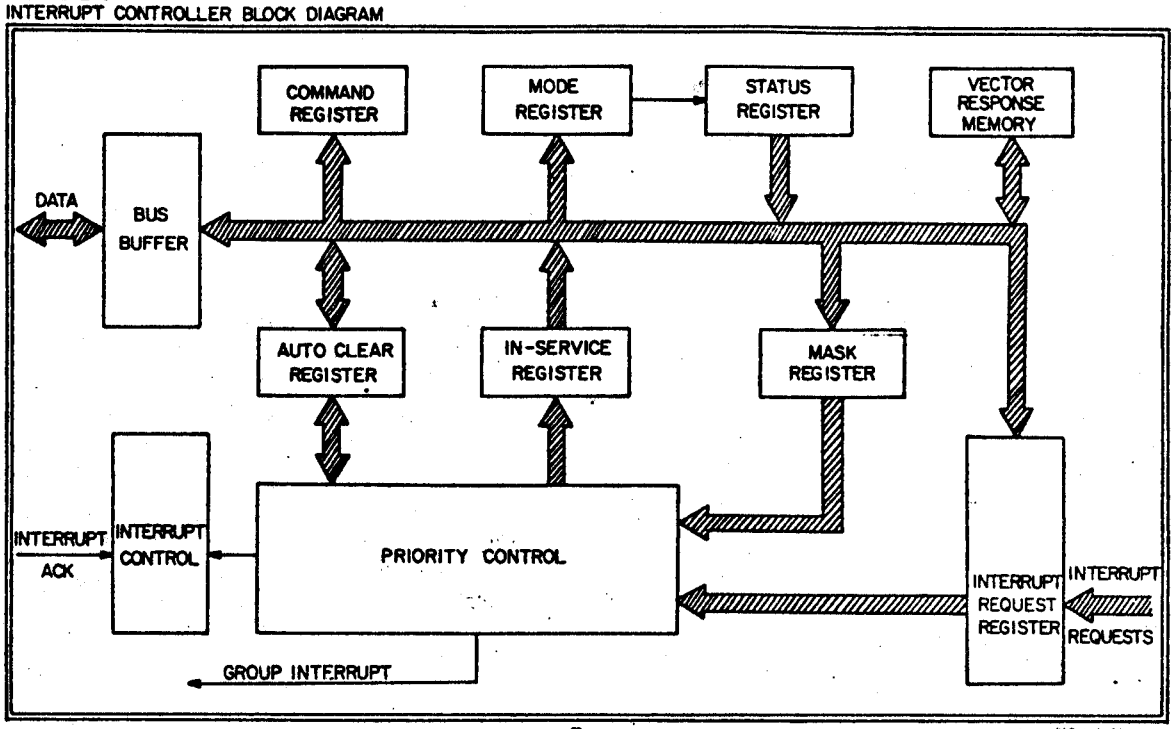
Note that the address table, ITAB, must be located in memory such that the 16 bit service addresses always start on even addresses, and that the least significant bit from the interrupt controller is always a 0. Figure 2-5 illustrates the operation of mode 2 interrupts.

2.5 Interrupt Controller Architecture

Figure 2-6 gives a block diagram of the interrupt controller used on the Series II Z-80. Interrupt requests are captured by the interrupt request register (IRR) where noise spikes are filtered out and the interrupt requests are latched. Any requests that are not masked by the interrupt mask register (IMR) are passed to the priority control circuitry where they

will cause a group interrupt to be generated. When the processor accepts the interrupt, it issues an interrupt acknowledge signal which causes (1) the priority of all pending interrupt requests to be resolved, and (2) the byte from the vector response memory associated with the highest priority request to be asserted on the processor's data bus. The Z-80 processor uses this response vector as a pointer to a table of service routine addresses, fetches the appropriate address from the table, and executes a subroutine call to that address.

FIGURE 2-6
INTERRUPT CONTROLLER BLOCK DIAGRAM



Other interrupt management functions are controlled by the Auto Clear register, the interrupt service register, and the mode register. All setup and programming of the interrupt controller is exercised through the command register, and the internal state of the controller is available in the status register.

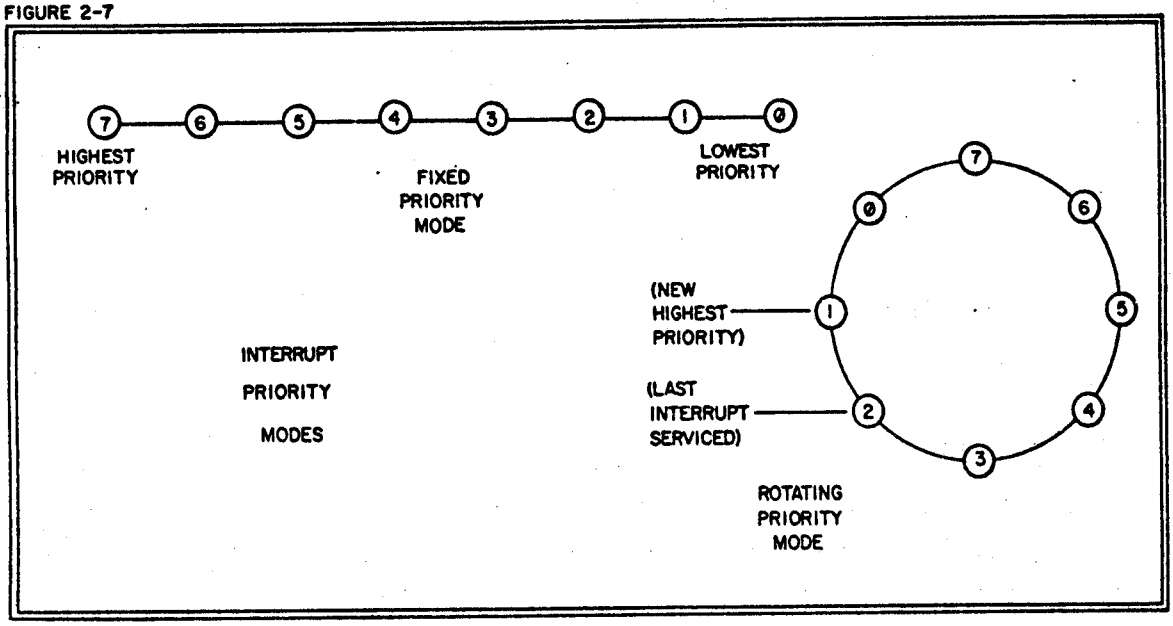
2.6 Interrupt Controller Operating Options

The interrupt controller's mode register is used to establish the basic operating conditions and options for constructing an interrupt driven system.

Fixed or Rotating Priorities

Bit M0 in the mode register specifies whether the interrupt controller operates in the fixed priority mode or in the rotating priority mode. The fixed priority mode (M0=0) assigns the eight interrupt request inputs the priority they are assigned on the S-100 bus, that is, the VI 7 line is the highest priority interrupt in the system and the VI 0 line is the least significant interrupt in the system. This is shown in Figure 2-7.

FIGURE 2-7



The fixed priority mode is the one most often used, although there exists a possibility that low priority interrupts may never be serviced in a heavily loaded system. The individual masking feature, however, may be used to modify the effective priority structure of the interrupt controller to guarantee service to all peripherals.

In the fixed priority mode interrupts are normally masked such that only an interrupt of a priority higher than the interrupt being serviced may generate a new group interrupt to the processor. With the Series II Z-80, each interrupt may be specified to erect such a masking "fence" or not to, by setting the corresponding bit in the Auto Clear register, allowing a great deal of flexibility in the effective system priorities of each interrupting device.

If the eight interrupts have similar priority and bandwidth requirements, one effective solution is to select the rotating priority option (M0=1). As shown in the figure, the relative priorities in the system remain the same, except that the chain is closed into a circle. In rotating priority mode, however, the lowest priority position is assigned by hardware to the last interrupt serviced. This rotating priority scheme prevents any single interrupt from dominating the system. It assures that any interrupt will

not have to wait more than seven interrupt cycles before being serviced. Note that there is no nesting of interrupts in the rotating mode; all pending interrupts are masked from the processor while an interrupt is being serviced.

Common and Individual Vectoring

Bit M1 in the mode register specifies whether the individual interrupts will branch to different locations upon acceptance of an interrupt, or whether all interrupts will branch to the same location. In the common vectoring mode, the response associated with IREQ0 (VI 7) will be asserted upon interrupt acknowledge, regardless of which interrupt is being acknowledged. The common vectoring mode is useful if all service routines save the total machine state before entering the actual device service routine. In such a case, the processor branches to the common register save routine, then polls the interrupt controller to discover the highest priority interrupt pending, and branches to the appropriate service procedure. This approach avoids duplication of the register save routine. Bit M1=0 selects the individual vectoring mode while M1=1 selects common vectoring.

Polled Mode

Bit 2 of the mode register allows the system to disable the group interrupt output from the interrupt controller. In the polled mode the processor may read the status register in the interrupt controller to see if any interrupt requests are pending, and which request has the highest priority. Interrupt request bits may be cleared by software. The polled mode option then bypasses the hardware interrupt, the vectoring and fencing functions of the interrupt controller while the request latching, masking, and priority resolution remain unaffected. Bit M2=0 selects the interrupt mode, M2=1 selects the polled mode.

Master Mask

Bit 7 of the mode register specifies the status of the master mask bit of the interrupt controller. When the master mask bit is set, (M7=0) the controller is disarmed just as if all the mask bits in the mask register had been set. When bit 7 is a 1, the chip is armed and any unmasked active requests will cause an interrupt output.

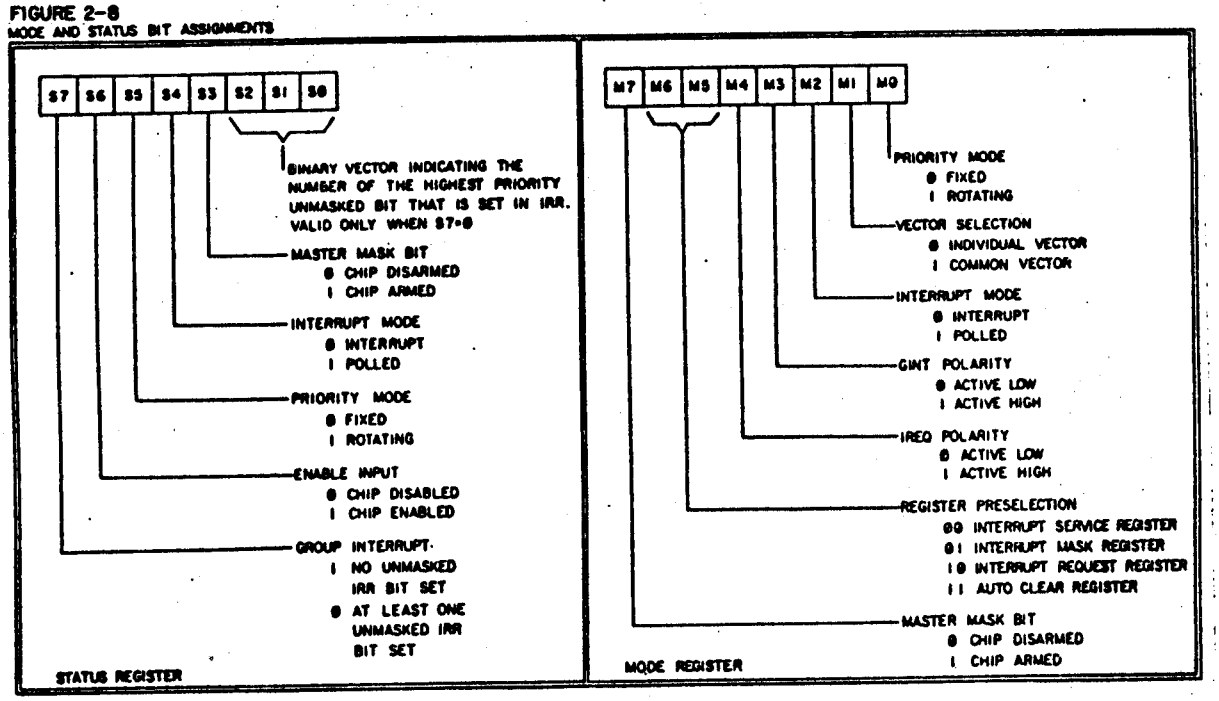
The remaining bits of the mode register specify (1) the polarity of the group interrupt output, which is always active low on the Series II Z-80 (M3=0); (2) the polarity of the interrupt request signals, which is always active low in S-100 systems (M4=0); and (3) the register pre-selection address. These bits select which of the internal registers will be read by

the processor during a data read operation from the controller. A summary of the data transfer to and from the controller is given in a later section.

Status Register

The Status register describes the internal state of the interrupt controller, as summarized in Figure 2-8. The 8 bits of the status register reflect the operating modes described above, and indicate the highest priority unmasked interrupt that is pending. Status bits S0-S2 contain the complement of the binary vector indicating the highest priority interrupt pending. The field should be considered invalid unless the status bit S7 (the group interrupt bit) is set (S7=1), indicating that at least one of the interrupt request bits is set and unmasked. Status bit S3 indicates the condition of the master mask in the Mode register. S3 equals 0 indicates that the chip is disarmed, S3 equals 1 indicates the chip is armed. Bit S4 indicates whether the controller is in interrupt mode or polled mode, and bit S5 indicates whether the priority resolution is in fixed or rotating mode. Finally, bit S6 is not used in this implementation.

FIGURE 2-8
MODE AND STATUS BIT ASSIGNMENTS



The Status register is read directly by executing a read from the controller's control port; it does not require any preselection.

Interrupt Service Register (ISR)

The ISR is eight bits long and is used to store the acknowledge status of the interrupt requests. When an interrupt is acknowledged, the controller clears the associated request bit and sets the corresponding bit in the ISR. When the ISR bit is programmed for automatic clearing, the ISR bit will be cleared before the end of the acknowledge sequence. If it is not set for auto-clear, the ISR bit must be cleared by command from the Z-80.

Internally, the controller uses the ISR to erect a "masking fence". When an ISR bit is set and the fixed priority mode is selected, only an interrupt of higher priority than the one being serviced will cause a new interrupt output, nesting the interrupt routines, and moving the masking fence up to the new level. New interrupt requests of a priority lower than the one being serviced will be masked until the ISR is cleared by the Z-80. When the Auto Clear is specified, no masking fence is erected, since the ISR bit is cleared during the acknowledge sequence. The Auto-Clear function may be specified for each interrupt individually, by setting the associated bit in the Auto Clear register (ACR).

Information Transfers

There are two input/output ports associated with the interrupt controller -- a control port and a data port. A read operation executed from the control port will always read the status register, and a write to the control port will always write into the command register. When a read or write is executed to the data port, however, the information transfered depends on which of the internal registers has been pre-selected by the preceding command. During read operations, the Interrupt Request Register (IRR), the Interrupt Service Register (ISR), the Interrupt Mask Register (IMR), or the Auto Clear Register (ACR) are pre-selected for reading by the bits M5 and M6 of the mode Register. For writing into these registers, specific commands must be issued to the controller; the M5 and M6 bits do not preselect registers for writing.

2.7 Commands to the Interrupt Controller

The interrupt controller's command set allows the Z-80 processor to select and alter all the operating modes described above, to customize the controller for different applications. Commands are entered into the command register by writing to the control port. In the commands described below, "X" indicates a "don't care" bit position.

RESET

Coding: 0 0 0 0 0 0 0 0

The reset command establishes a known internal condition in the interrupt controller:
Mode = Fixed priority, individual vectors, interrupt (non-polled) operation, interrupt inputs and output in the active low sense (normal), ISR preselected for reading, chip disarmed by master mask.

IMR = all ones, all requests are masked.
ACR = all zeros, no auto clear specified.
ISR and IRR are cleared of all old requests.

CLEAR IRR AND IMR

Coding: 0 0 0 1 0 X X X

All bits in the Interrupt Request Register and the Mask Register are cleared, no interrupts pending, no interrupts masked.

CLEAR SINGLE IMR AND IRR BIT

Coding: 0 0 0 1 1 B2 B1 B0

A single bit is cleared in the IMR and the IRR. The bit is specified by the three bit field B2-B0 where:

B2	B1	B0	BIT SPECIFIED
0	0	0	0 (LSB)
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7 (MSB)

CLEAR IMR

Coding: 0 0 1 0 0 X X X

All bits in the IMR are cleared to zeros, thus all interrupt requests are unmasked.

CLEAR SINGLE IMR BIT

Coding: 0 0 1 0 1 B2 B1 B0

A single bit in the mask register is cleared, as specified in the field B2-B0.

SET IMR

Coding: 0 0 1 1 0 X X X

All bits in the mask register are set to ones, thus masking all interrupt requests.

SET SINGLE IMR BIT

Coding: 0 0 1 1 1 B2 B1 B0

A single mask bit is set, corresponding to the bit specified in the three bit field. The corresponding interrupt is then masked out of the priority resolution circuitry.

CLEAR IRR

Coding: 0 1 0 0 0 X X X

All bits in the request register are cleared to zeros. New transitions on the interrupt inputs will be necessary to cause an interrupt.

CLEAR SINGLE IRR BIT

Coding: 0 1 0 0 1 B2 B1 B0

A single request bit is cleared to zero, specified by B2-B0.

SET IRR

Coding: 0 1 0 1 0 X X X

All the request bits are set to ones, any unmasked bit will cause an interrupt output. Useful in testing.

SET SINGLE IRR BIT

Coding: 0 1 0 1 1 B2 B1 B0

A single request bit is set, corresponding to the bit specified by B2-B0.

CLEAR ISR

Coding: 0 1 1 1 0 X X X

All Service register bits are cleared to zero, fencing is eliminated.

CLEAR HIGHEST PRIORITY ISR BIT

Coding: 0 1 1 0 X X X X

The highest priority bit in the service register is cleared. Caution should be exercised in using this command if the auto clear option is specified, since the bit cleared by this command may not be the bit intended.

CLEAR SINGLE ISR BIT

Coding: 0 1 1 1 1 B2 B1 B0

The bit specified by B2-B0 is cleared in the service register.

LOAD MODE BITS M0 THROUGH M4

Coding: 1 0 0 M4 M3 M2 M1 M0

The five low order bits of the mode register are loaded. This command controls all operating options except the Master Mask and read preselection.

CONTROL MODE BITS M5,M6,M7

Coding: 1 0 1 0 M6 M5 N1 N0

The field in the command is loaded into the register preselect bits in the mode register. The N1, N0 field in the command controls the master mask as follows:

N1	N0	
0	0	NO CHANGE TO M7
0	1	SET M7
1	0	CLEAR M7
1	1	(illegal)

PRESELECT IMR FOR WRITE

Coding: 1 0 1 1 X X X X

The next write to the controller's data port will load the mask register. Various read operations may be inserted between this command and the write operation.

PRESELECT ACR FOR WRITE

Coding: 1 1 0 0 X X X X

The auto clear register will be loaded on the next write to the data port.

PRESELECT RESPONSE MEMORY FOR WRITE

Coding: 1 1 1 0 0 L2 L1 L0

The vector response memory is preselected for writing, the byte which will serve as a pointer to the Z-80's service jump table will be loaded on the next write to the data port into the level specified by L2-L0 where:

L2	L1	L0	BUS INTERRUPT (VI #)
1	1	1	VI 0
1	1	0	VI 1
1	0	1	VI 2
1	0	0	VI 3
0	1	1	VI 4
0	1	0	VI 5
0	0	1	VI 6
0	0	0	VI 7

2.8 Using the Interrupt Controller

Before the Interrupt Controller can do useful work it must be initialized under control of the Z-80 processor. Because of the many operating options available with the interrupt controller, the controller may be programmed using many different approaches. The following is but one of these ways to construct a basic initialization and service routine.

1) Disable Processor interrupts
2) Reset the interrupt controller
3) Setup vector response memory
4) Load operating options into the mode register
5) Load the mask conditions into the IMR
6) Clear the IRR
7) Clear the master mask
8) Set processor interrupt mode to mode 2, and load the service jump table.
9) Enable interrupts.

The following simple program uses the Z-80 interrupt mode 2, and is intended to both serve as a test of the interrupt controller and an example program for using the interrupt controller.

```
;       VECTORED INTERRUPT CONTROLLER HANDLERS
;
;THE FOLLOWING ROUTINES PROVIDE THE NECESSARY PROGRAMMING
;SUPPORT TO SETUP AND RESPOND TO THE EIGHT LEVELS OF VECTORED
;INTERRUPTS ON THE S-100 BUS.
;THE ROUTINES SUPPORT BOTH INTERRUPT DRIVEN AND POLLED
;OPERATION, FIXED AND ROTATING PRIORITIES, ARBITRARY
;INDIVIDUAL MASKING, AND ARBITRARY AUTO-CLEAR FOR NESTING
;CONTROL.
;
;ALL REGISTERS IN THE MAIN REGISTER BANK OF THE Z-80 CPU ARE
;SAVED UPON ENTRY TO AN INTERRUPT PROCEDURE, AND RESTORED
;UPON EXIT FROM THE INTERRUPT.
;
;DEFINITIONS
;
IOBASE  EQU     0E0H    ;BASE I/O ADDRESS ON Z-80 BOARD
ICON    EQU     IOBASE+1;9519 CONTROL PORT
IDAT    EQU     IOBASE  ;9519 DATA PORT
FP      EQU     0FFH    ;FRONT PANEL LIGHTS (OUTPUT) AND
                        ;SWITCHES (INPUT) FOR TEST ROUTINE
STK     EQU     1000H   ;STACK POINTER FOR TEST ROUTINE
RESTRT  EQU     0D049H  ;PUT EXIT FROM TEST ROUTINE HERE.
;
;ENTRY POINTS
        ORG     0100H
TEST:   JMP     TEST0   ;INTERRUPT TEST.
POLTST: JMP     PLTST0  ;POLLED TEST.
SETUP:  JMP     SETUP0
MODE:   JMP     MODE1
POLL:   JMP     POLL0
SERV:   JMP     SERV0
SAVE:   JMP     SAVE0
EXIT:   JMP     EXIT0
;
;VARIABLES:
;
MODES:  DB      00      ;00 = FIXED PRIORITY & INT MODE
                        ;01 = ROTATING PRIORITY & INT MODE
                        ;04 = FIXED PRIORITY & POLLED MODE
                        ;05 = ROTATING PRIORITY & POLLED MODE
MASKS:  DB      00      ;A 1 WILL MASK THE INTERRUPT IN THE
                        ;BIT POSITION: DATA0=VI7,DATA7=VI0
CLEARS: DB      00      ;A 1 IN A BIT POSITION WILL AUTO CLEAR
                        ;THE CORRESPONDING INT: DATA0=VI7 ETC.
;
```

```
;THE LOCATIONS OF THE INTERRUPT SERVICE ROUTINES ARE STORED IN
;THE JUMP TABLE BELOW, ITAB.  CARE MUST BE TAKEN WHEN ORG'ING
;THE PROGRAM SUCH THAT ITAB NEVER OVERFLOWS A 256 BYTE BOUNDARY
;IN MEMORY, AND THAT ITAB BEGINS AT AN EVEN MEMORY LOCATION.
;
        DB      0       ;PSEUDO-OPS SO ITAB GOES
NOW     SET     $/2     ;AT RIGHT PLACE.
        ORG     NOW*2   ;=EVEN ADDRESS.
TEMP0   SET     $+0EH   ;LAST LOCATION OF TENTATIVE ITAB.
TEMP1   SET     TEMP0&0FF00H    ;ISOLATE 2 MSBs.
TEMP2   SET     $&0FF00H        ;IF THEY'RE THE SAME,
        IF      TEMP1-TEMP2     ;DON'T ASSEMBLE THIS CODE.
        ORG     TEMP1           ;OTHERWISE, USE BOUNDARY
        ENDIF                   ;FOR ORIGIN.


ITAB:   DW      INT0    ;JMP LOCATION FOR VI 7 ROUTINE
        DW      INT1    ;  CORRESPONDS TO LEVEL 0 AT
        DW      INT2    ;  VECTORED INTERRUPT CONTROLLER
        DW      INT3    ;  AND SO ON.
        DW      INT4
        DW      INT5
        DW      INT6
        DW      INT7
;
```

```
;SETUP SUBROUTINE:  THE SETUP SUBROUTINE INITIALIZES THE 9519,
;LOADS THE OPERATING OPTIONS DESCRIBED IN THE VARIABLES, LOADS
;THE RESPONSE MEMORY WITH VECTORS TO THE JUMP TABLE (ITAB), AND
;SETS THE Z-80 TO INTERRUPT MODE 2.
;
SETUP0: DI              ;DISABLE PROCESSOR INTERRUPTS
        IM2             ;SET INTERRUPT MODE 2
        LXI     H,ITAB  ;HL POINTS TO THE JUMP TABLE
        MOV     A,H     ;HIGH BYTE TO ACCUMULATOR.
        STAI            ;I REGISTER CONTAINS HIGH ORDER PNTR
        XRA     A       ;A=0
        OUT     ICON    ;RESET 9519
        CALL    MODE0   ;SET THE CONTROLLER OPERATING MODES

        MVI     B,0E0H  ;COMMANDS E0-E7 PRESELECT MEMORY LEVEL
        MVI     C,8     ;LEVEL COUNTER = 8
LDMEM:  MOV     A,B     ;A = PRESELECT NEXT LEVEL COMMAND
        OUT     ICON    ;PRESELECT NEXT LEVEL
        MOV     A,L     ;A = LOW PNTR TO JUMP TABLE
        OUT     IDAT    ;WRITE THIS MEMORY LEVEL
        INR     L
        INR     L       ;JUMP TABLE PNTR = NEXT ENTRY
        INR     B       ;NEXT LEVEL.
        DCR     C       ;DECREMENT LEVEL COUNTER
        JNZ     LDMEM   ;DONE FILLING MEMORY?

        MVI     A,040H
        OUT     ICON    ;CLEAR IRR
        MVI     A,070H
        OUT     ICON    ;CLEAR ISR
        MVI     A,0A9H
        OUT     ICON    ;CLEAR MASTER MASK PRESELECT IRR
        EI              ;ENABLE PROCESSOR INTERRUPTS
        RET             ;EXIT

MODE0:  LDA     MODES   ;A=MODE
        ORI     80H     ;MAKE INTO MODE SET COMMAND
        OUT     ICON    ;SET MODE

        MVI     A,0B0H  ;A = PRESELECT MASK COMMAND
        OUT     ICON    ;PRESELECT THE MASK REGISTER FOR WRITE
        LDA     MASKS   ;A = MASK
        OUT     IDAT    ;WRITE THE MASK

        MVI     A,0C0H  ;A = AUTO CLEAR PRESELECT COMMAND
        OUT     ICON    ;PRESELECT ACR
        LDA     CLEARS  ;A = AUTO CLEAR PATTERN
        OUT     IDAT    ;WRITE AUTO CLEAR REGISTER
        RET             ;EXIT
;
```

```
;MODE1 IS A SUBROUTINE WHICH CHANGES THE OPERATING MODES TO THE
;VALUES CONTAINED IN THE VARIABLES AND REINITIALIZES THE
;CONTROLLER.  IT IS TO BE USED DYNAMICALLY.  CHANGE THE VALUES
;AT MODES, MASKS, AND CLEARS AND CALL MODE.  THIS WILL CHANGE
;THE OPERATING OPTIONS WITHOUT AFFECTTING IRR OR ISR.
;
MODE1:  DI              ;DISABLE INTERRUPTS
        CALL    MODE0   ;SET MODES
        EI              ;ENABLE INTERRUPTS
        RET             ;EXIT
;
;THE FOLLOWING SUBROUTINES ARE USED IF THE POLLED MODE
;OF OPERATION HAS BEEN SELECTED.  A CALL TO POLL WILL RETURN
;THE CARRY BIT CLEAR IF AN INTERRUPT REQUEST IS PENDING.
;OTHERWISE THE CARRY WILL BE SET.
;A CALL TO SERV WILL SERVICE THE HIGHEST PRIORITY INTERRUPT
;THAT IS PENDING IN THE SYSTEM AND RETURN.
;POLL USES A & F
;SERV USES A,F,DE,HL
;
POLL0:  IN      ICON    ;INPUT CONTROLLER STATUS BYTE
        RLC             ;PUT INTERRUPT PENDING FLAG IN CARRY
        RET             ;EXIT

SERV0:  IN      ICON    ;INPUT STATUS BYTE
        ANI     07H     ;MASK FOR LEVEL
        RLA             ;OFFSET=LEVEL*2
        LXI     H,ITAB  ;HL POINT TO JUMP TABLE
        ADD     L       ;HL+OFFSET=VECTOR
        MOV     L,A     ;HL=VECTOR
        MOV     E,M
        INX     H
        MOV     D,M     ;DE=ROUTINE ADDRESS
        XCHG            ;HL=DE
        PCHL            ;JUMP TO SERVICE ROUTINE
;
```

```
;ROUTINES SAVE AND EXIT:  THESE ROUTINES ARE INTERRUPT SERVICE
;SUPPORT ROUTINES.  THE SAVE ROUTINE SAVES THE MAIN REGISTER
;BANK IN THE PROCESSOR CHIP, REENABLES INTERRUPTS, AND RETURNS
;TO THE SERVICE ROUTINE.  IT MUST BE CALLED AT THE BEGINNING
;OF EVERY SERVICE ROUTINE --
;       START:  CALL    SAVE
;THE EXIT ROUTINE SIGNALS THE INTERRUPT CONTROLLER THAT THE
;SERVICE ROUTINE IS FINISHED, RESTORES THE MACHINE STATE, AND
;RETURNS TO THE INTERRUPTED PROGRAM, OR IN THE CASE OF POLLED
;OPERATION, THE CALLING PROGRAM.  EXIT  SHOULD BE ENTERED BY A
;JMP INSTRUCTION, WITH THE LEVEL OF THE INTERRUPT BEING
;SERVICED IN THE A REGISTER.  SERVICE ROUTINES SHOULD BE
;STRUCTURED, THEN:
;       START:  CALL    SAVE
;               ...
;       (MAIN SERVICE ROUTINE)
;               ...
;       MVI     A,#     ;# EQUALS THE SERVICE LEVEL
;       JMP     EXIT    ;RESET 9519 & RESTORE STATE
;
SAVE0:  XTHL            ;HL GETS (SP)=PC
                        ;(SP) GETS HL
        PUSH    PSW     ;SAVE A
        PUSH    B       ;SAVE BC
        PUSH    D       ;SAVE DE
        EI              ;ENABLE INTERRUPTS
        PCHL            ;PC GETS HL (RETURNS)
EXIT0:  DI              ;DISABLE INTERRUPTS
        ORI     078H    ;CREATE CLEAR ISR BIT COMMAND
        OUT     ICON    ;CLEAR ISR BIT
        ANI     07H     ;RESTORE LEVEL
        ORI     048H    ;CREATE CLEAR IRR BIT COMMAND
        OUT     ICON    ;CLEAR IRR BIT
        POP     D
        POP     B
        POP     PSW
        POP     H       ;RESTORE MACHINE STATE
        EI              ;ENABLE INTERRUPTS
        RET             ;DONE WITH SERVICE
;
```

```
;TEST PROGRAM:  THESE TEST PROGRAMS USE THE ABOVE ROUTINES TO
;TEST THE OPERATION OF THE CONTROLLER IN A SYSTEM. THE ROUTINES
;LIGHT A BIT ON THE PROGRAMMED OUTPUT PORT OF THE
;INTERSYSTEMS FRONT PANEL CORRESPONDING TO THE MOST RECENT
;INTERRUPT SERVICED. THE S-100 VI 7 LINE CORRESPONDS TO A
;LEVEL 0 INTERRUPT AT THE INTERRUPT CONTROLLER, AND SHOULD
;LIGHT FRONT PANEL PORT LIGHT 0. A SIMPLE TEST OF THE
;INTERRUPT SYSTEM IS TO RUN TEST AND GROUND EACH VI LINE IN
;TURN, CHECKING THE PROPER FRONT PANEL LIGHT.
;THE TEST ROUTINES EXIT TO RESTRT WHEN THE MOST SIGNIFICANT
;FRONT PANEL SWITCH IS SENSED LOW. PLTST0 IS THE POLLED
;VERSION OF THE TEST ROUTINE WHICH OPERATES SIMILARLY.
;THE STATE OF THE NEXT-TO-MOST-SIGNIFICANT FRONT PANEL
;SWITCH DETERMINES WHICH ROUTINE RUNS -- HIGH = INTERRUPT TEST,
;LOW  = POLLED TEST.
;
TEST0:  LXI     SP,STK  ;SET STACK POINTER
        XRA     A       ;ACCUMULATOR GETS 0.
        OUT     FP      ;CLEAR FRONT PANEL LIGHTS
        STA     MODES   ;INTERRUPTS AND FIXED PRIORITY.
        CALL    SETUP   ;SETUP CONTROLLER.
HALT:   IN      FP      ;GET SWITCHES.
        RLC             ;PUT D7 INTO CARRY,
        JNC     RESTRT  ;AND EXIT IF LOW.
        RLC             ;PUT D6 INTO CARRY.
        JNC     POLTST  ;DO OTHER TEST IF D6 IS LOW.
        HLT             ;HALT & WAIT FOR INTERRUPT
        JMP     HALT    ;CONTINUE FOREVER

PLTST0: LXI     SP,STK
        MVI     A,4
        STA     MODES   ;=FIXED PRIORITY, POLLED MODE
        CALL    SETUP
        XRA     A       ;A = 0.
        OUT     FP      ;BLANK LIGHTS.
PLTST1: IN      FP      ;GET SWITCHES.
        RLC             ;PUT D7 INTO CARRY.
        JNC     RESTRT  ;EXIT IF SWITCH IS LOW. OTHERWISE,
        RLC             ;PUT D6 INTO CARRY.
        JC      TEST    ;GO TO OTHER TEST IF SWITCH IS UP.
        CALL    POLL    ;CHECK IF ANYTHING NEW.
        CNC     SERV    ;YES? GO DO IT.
        JMP     PLTST1  ;REPEAT.
;
```

```
;INTERRUPT SERVICE ROUTINES:
;
INT0:   CALL    SAVE    ;SAVE ENVIRONMENT FIRST.
        MVI     A,01H   ;A GETS LIGHT PATTERN
        MVI     B,0     ;B GETS LEVEL #
        JMP     INT     ;BRANCH TO COMMON SERVICE ROUTINE
INT1:   CALL    SAVE
        MVI     A,02H
        MVI     B,1
        JMP     INT
INT2:   CALL    SAVE
        MVI     A,04H
        MVI     B,2
        JMP     INT
INT3:   CALL    SAVE
        MVI     A,08H
        MVI     B,3
        JMP     INT
INT4:   CALL    SAVE
        MVI     A,10H
        MVI     B,4
        JMP     INT
INT5:   CALL    SAVE
        MVI     A,20H
        MVI     B,5
        JMP     INT
INT6:   CALL    SAVE
        MVI     A,40H
        MVI     B,6
        JMP     INT
INT7:   CALL    SAVE
        MVI     A,80H
        MVI     B,7
        JMP     INT

INT:    OUT     FP      ;UPDATE LIGHTS
        MOV     A,B     ;A GETS LEVEL #
        JMP     EXIT    ;RESTORE MACHINE STATE
        END
;
```

2.9 Series II Bus Interface

The Series II Z-80 provides a number of operating options which give the user some control over the processor-to-bus interface, while still corresponding to the IEEE bus specification. These options include processor speed selection of 2 or 4 MHz and optional addition of a single wait state to all instruction fetch cycles (the instruction fetch cycle of the Z-80 processor is shorter than other memory cycles of the processor, and represents the worst case for memory access times), all memory cycles, or all I/O cycles.

In addition to these basic options a special feature has been included to allow the implementation of extremely high reliability systems. In order to meet the IEEE standard for S-100 bus devices using a Z-80 processor, all bus address and status signals should be latched. This is because the Z-80 chip can change its effective address at the end of an instruction fetch cycle while the bus read strobe is still valid. The IEEE specification correctly prohibits such timing ambiguities. To comply with this specification the Series II Z-80 includes hardware latches on all address and status lines to the S-100 bus.

There are two modes of operation of these latches: Partial Latching and Full Latching. The Full Latch mode is designed for extremely high reliability systems, or operation in electrically noisy environments. The Full Latch mode halves the number of signal transitions on the bus and restricts all changes in address and status signals to specific, non-critical parts of the bus cycle, thus drastically reducing bus noise and signal crosstalk.

The Full Latch mode decreases the effective memory access time of the processor, requiring that either very fast memory boards be used (160 nsec. worst case board access for M1 cycles requires chip speed of approx. 125 nsec in a typical design), or that a single wait state be added to all M1 cycles, slowing the processor approximately 10%.

Other system parameters will usually reduce the wait state requirement to insignificance, but the occasional necessity for the faster M1 cycle can be satisfied with the Partial Latch mode. This mode does not control signal transitions as does the Full Latch mode, but provides for operation at 4 mHz without wait states if proper cycle triggerring is employed. All InterSystems Series II memory boards will support no-wait operation at 4 MHz using the Partial Latch mode.

2.10 Direct Memory Access Operations

The last major feature of the Series II Z-80 concerns the relationship of the processor to Direct Memory Access devices on the bus.

The IEEE bus specification defines a special protocol for the transfer of bus control from a permanent bus master, in this case your Series II Z-80, to a temporary bus master such as a DMA device or attached peripheral processor. This protocol involves a specially timed and overlapped transfer of the various signal groups on the bus such that the DMA device and the CPU are both driving the most critical bus lines in given inactive states during the transfer operation.

The circuit which controls this timing could be included on either the processor board or on the DMA board. It has been included on the Series II Z-80 board for two reasons: First, if it is included on the processor board, it need not be duplicated on every DMA board in the system, and second, since no currently existing DMA boards include such a circuit, it is much easier to modify an existing board to meet the S-100 standard if this circuit is centralized on the processor board.

The Series II Z-80 board, then, contains a circuit which will conduct all the timing of the bus transfer, and tri-state its own bus drivers according to the IEEE specification (sections 2.8.2.1 through 2.8.2.4). DMA devices may sense the two bus transfer signals, XFER I and XFER II, on the SDSB line and the CDSB line respectively.

The Z-80 also includes pull-up resistors on the four DMA arbitration lines, so that they need not appear elsewhere in the system, and 10 ohm resistors in series with the control output lines, to prevent driver fatigue and glitches during the overlapped bus transfer.

Section 3.0

Board Setup

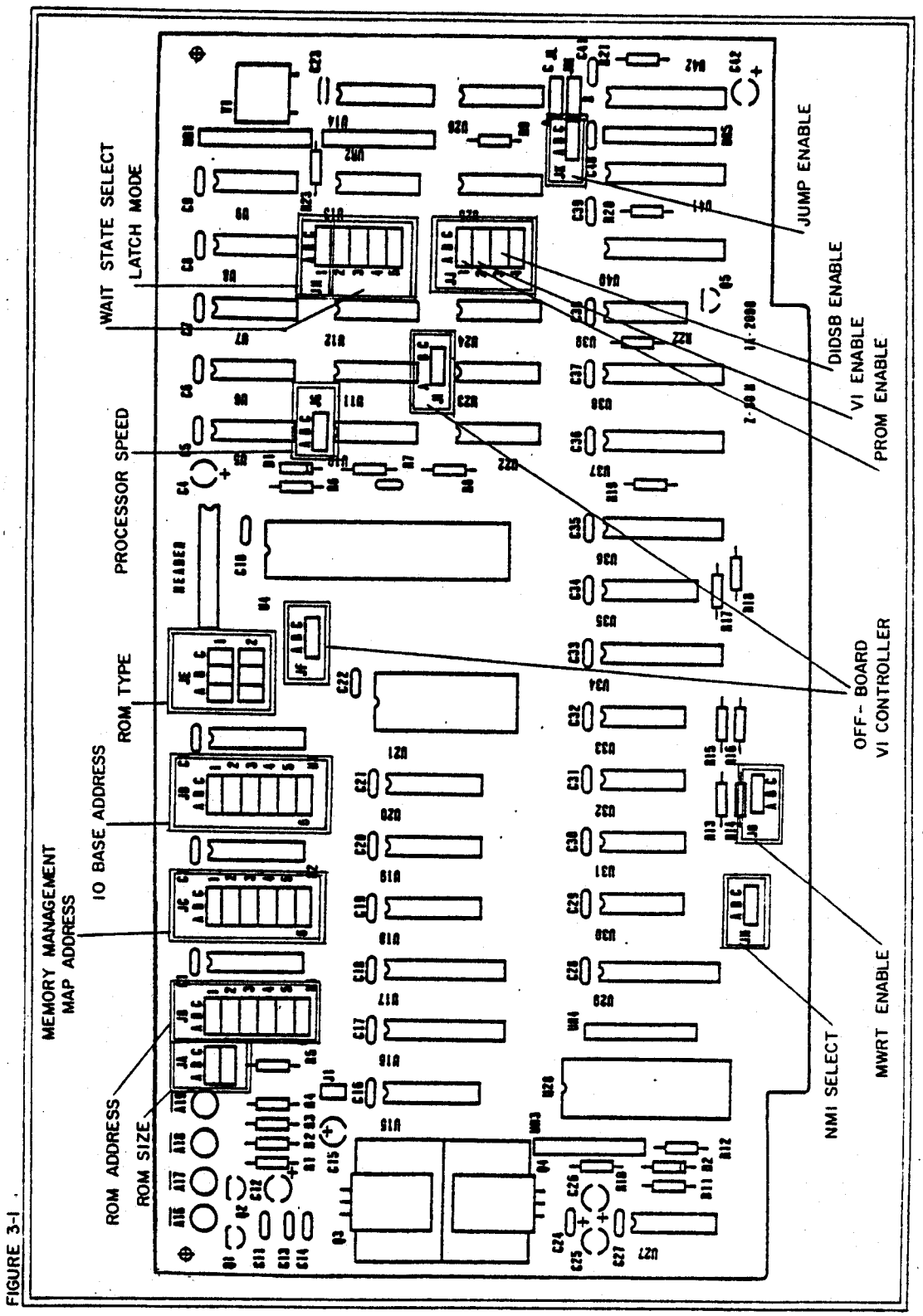

MEMORY MANAGEMENT MAP ADDRESS
IO BASE ADDRESS
ROM TYPE
PROCESSOR SPEED
LATCH MODE
WAIT STATE SELECT
ROM ADDRESS
ROM SIZE
JUMP ENABLE
DISB ENABLE
VI ENABLE
PROM ENABLE
OFF- BOARD VI CONTROLLER
NMI SELECT
MWRT ENABLE

FIGURE 3-1

3.0 Board Setup

This section of the manual describes how to set up the Series II Z-80 for operation in your system. Certain jumpers on your board are only used in assembly or in testing, and these will be described only in the technical reference section.

Figure 3-1 shows the board location of the jumpers we are concerned with in this section.

3.1 Bus Interface Setup

The first and most important choice in the setup of the Z-80 board is the selection of processor speed. Either 2 MHz or 4 MHz operation may be selected with the pin-jumper JG. The choice of processor speed is a complicated decision involving the access times of the other boards in the system, the latch control mode selected, and the use of wait states.

Table 3-1 gives the various access times associated with the different types of processor cycles at 2 and 4 MHz, for the full and partial latching modes. Note also that the effective access time differs for static and dynamic or edge-triggered devices. This is because an edge-activated device such as a dynamic memory must wait to begin its cycle until certain information is guaranteed to be valid, while static devices respond to address changes as fast as possible, and do not require a strobe signal to begin their cycle.

Table 3-1 Processor Access Times (in nsec.)

Cycle Type	2 MHz	2 w/wait	4 MHz	4 w/wait
M1 Stat. Platch	780	1280	280	530
M1 Stat. Flatch	570	1070	220	470
M2 Stat. Platch	1020	1520	395	645
M2 Stat. Flatch	810	1310	310	560
M1 Dyn. Platch	580	1080	220	470
M1 Dyn. Flatch	410	910	160	410
M2 Dyn. Platch	810	1310	310	560
M2 Dyn. Flatch	640	1140	270	520
I/O F or Platch	900	1400	400	650

Where: All times are in nanoseconds, M1 refers to an instruction fetch cycle, M2 refers to all other memory cycles, Platch refers to the partial latching mode, and Flatch refers to the full latch mode.

It should be noted that these are board level access times, rather than chip level access times, that is, no account has been taken of logic or buffer

delays on the memory board but full worst case delays on the processor have been included in the calculations.

It can be seen that the correct choice of bus cycle depends on a variety of factors. The most reliable bus cycle is the fully latched cycle, but access times are considerably shorter, especially for M1 cycles which are already short in the Z-80. It is therefore recommended that if the full latch mode is used at 4 MHz, a wait state be added to all M1 cycles. This will slow the operation of the processor approximately 10%, but allows the use of 300 nsec. static memory boards and 250 nsec. dynamic memory boards with ease. Adding a wait state to all memory cycles and using the full latch mode allows the use of less expensive 450 nsec. static or 400 nsec. dynamic memory boards, but slows the processor by 25%.

Full speed 4 MHz operation should use the partial latch mode. While this mode creates somewhat more bus noise than the full latch mode, it allows the use of 250 nsec. static memory boards, and 200 nsec. dynamic or edge-activated memories without wait states. (Note: The above calculations for edge-activated and dynamic memory boards in the partial latch mode are derived using the rising edge of the PSYNC signal as an address valid strobe to start the memory cycle, aborting the cycle later if it is an I/O cycle. In the full latch mode the start cycle trigger is the IEEE address and status strobe, STVAL.)

Selecting the Processor Speed

Pin-jumper JG selects the processor speed. It is located on the right side of the board between U5 and U10.

Processor Speed, JG	
2 MHz.	B to C
4 MHz.	A to B

Selecting the Latch Mode

The latch mode of the processor is selected at pin-jumper JH-1, located on the right side of the board between U12 and U13.

Latch Mode,	JH-1
Full latch	B to C
Partial latch	A to B

Adding Single Wait States

A single wait state may be added to any instruction fetch, any memory cycle (including instruction fetch cycles), any I/O cycle, and any reference to the on-board EPROM. The addition of these wait states is controlled by pin-jumpers JH-2 through JH-5, also located between U12 and U13. It should be noted that, for 4 MHz operation, all standard EPROM components will require a wait state for correct operation.

Wait State Selection, JH-2 thru JH-5

Cycle Type	Jumper	1 wait	no-wait
M1	JH-2	A to B	B to C
All Memory	JH-3	A to B	B to C
All I/O	JH-4	A to B	B to C
On-board EPROM	JH-5	A to B	B to C

Non-Maskable Interrupt Setup

The last setup we are concerned with in the bus interface section is the connection of the non-maskable interrupt line. This line, as yet unused in most S-100 systems, may be connected to either the NMI line on the S-100 bus, or to the Power Fail Pending line, or disconnected from the system.

If you are using a Power Fail anticipation circuit, and support the circuit with software routines that respond to an NMI as a power fail, then connect NMI to Power Fail.

If you have some other use for the Non-Maskable Interrupt, then it should be connected to its own pin on the bus. If you do not use the line in your system, disconnect the NMI line from the bus entirely by removing the shunt from the JN pin-jumper.

Non-Maskable Interrupt, JN

NMI to NMI	A to B
NMI to Power Fail	B to C
No NMI to Bus	OPEN

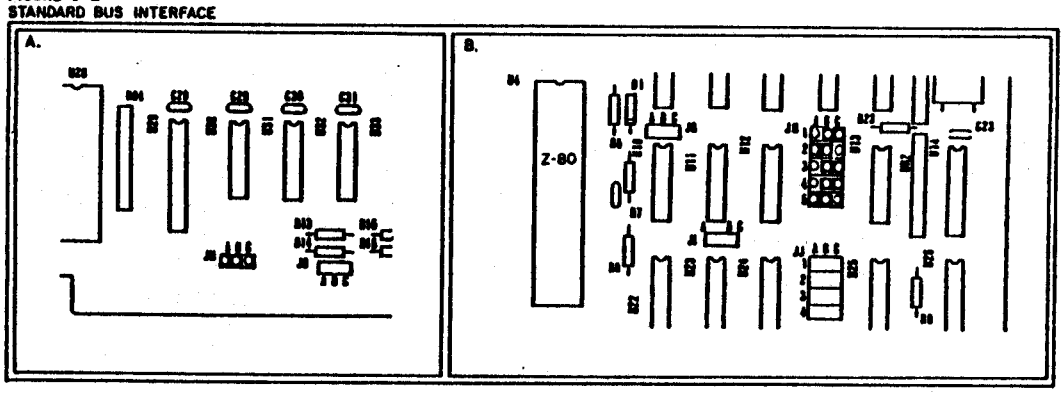
InterSystems Standard Bus Interface Configuration

Figure 3-2 shows the standard configuration for the bus interface as the board is shipped from the factory. This configuration represents the following options:

* 4 MHz operation
* Full latch mode
* Single wait for M1 and on-board EPROM
* NMI connected to NMI

Note that while the wait for the on-board EPROM is enabled, a wait state will not be added to the cycle until the EPROM is enabled with the EPROM enable jumper, to be described later.

FIGURE 3-2
STANDARD BUS INTERFACE



3.2 The On-Board EPROM and Automatic Bootstrap

The on-board EPROM socket may be configured to accept any of the following standard parts, or any compatible parts:

EPROM TYPES

Manufacturer	Part #	Size	Addressed at
Intel	2708	1 K	1 K boundary
	2758	1 K	1 K
	2716	2 K	2 K
	2732	4 K	4 K

The on-board EPROM and the reset jump functions are addressed at the same location in memory, which may be at any 1, 2, or 4 K boundary (depending on jumper set-up) in the 64k primary address space. When a EPROM is in the system and enabled, and the Reset jump is enabled, a reset to the computer system will be accompanied by a jump to the first byte of the on-board EPROM. This operation is referred to as the "auto-bootstrap". If the EPROM is disabled and the Reset Jump is enabled, the processor will jump to the location in main memory that is set at the EPROM address jumpers.

EPROM Type Select

The following table gives the settings for the EPROM type and size jumpers. Jumpers JA and JB are in the upper left hand corner of the board, and JE is in the up-center region. The "NONE" row gives the proper settings for using the Reset Jump function without using the on-board EPROM. The "STANDARD" row gives the settings for using the InterSystems' 2708 bootstrap EPROM at address F000 hex.

EPROM Type Select Jumpers

TYPE	JA-1	JA-2	JB-5	JB-6	JE-1	JE-2
2708	AB	AB	*	*	A	A
2758	AB	AB	*	*	B	B
2716	AB	BC	*	BC	C	B
2732	BC	BC	BC	BC	C	C
	(NO EPROM, JUMP TO ANY 1 K BOUNDARY)					
NONE	AB	AB	*	*	C	C
	(STANDARD= 2708 AT F000 HEX)					
STANDARD	AB	AB	BC	BC	A	A

Where "*" indicates that the jumper participates in the address selection for this EPROM type.

EPROM and Jump Address Selection

Once the EPROM type has been selected, it may be addressed in the 64 K primary address space of the processor using the jumpers at JB that were not previously assigned in type selection. The Jump and EPROM are addressed by defining a matching address for the starting location with jumpers JB. Connecting A to B will match a "one" on the corresponding address bit, and connecting B to C will match a "zero". The standard location for InterSystems bootstrap EPROMs is F000 hex. The Std. column gives the jumper settings for use with the standard boot EPROM.

EPROM and Jump Addressing

Jumper	Address Bit	"One"	"Zero"	Std.
JB-1	A 15	AB	BC	AB
JB-2	A 14	AB	BC	AB
JB-3	A 13	AB	BC	AB
JB-4	A 12	AB	BC	AB
JB-5*	A 11	AB	BC	BC
JB-6**	A 10	AB	BC	BC

* Not used in the addressing of 2732 EPROMs.
** Not used in the addressing of 2732 or 2716 EPROMs.

Enabling the EPROM and Jump

Once the EPROM type and address have been selected, the EPROM may be enabled by appropriate connection of the EPROM enable jumper JJ-1, located on the right side of the board between U 24 and U 25. If you do not wish to use the on-board EPROM, or if you wish to use the Reset jump without the on-board EPROM, JJ-1 should be used to disable the EPROM.

The Reset jump is enabled with jumper JK, located on the right side of the board just below U 25. The following table gives the different functions associated with the settings of these jumpers.

EPROM and Jump Enables

Function	JJ-1	JK
No EPROM, No Jump	BC	AB
EPROM enabled, No Jump	AB	AB
Auto-Boot (EPROM and Jump enabled)	AB	BC
Jump to Off-board memory	BC	BC

EPROM Wait States for 4 MHz Operation

It should be noted again that for 4 MHZ operation all currently available EPROM's of the 2708 family used on the Series II Z-80 will require a wait state added to any memory cycle which references the EPROM. A pin-jumper to add a wait state to EPROM cycles only has been described above. If this jumper, JH-5, is connected from A to B, and the EPROM is enabled, a single wait state will be added to all references to the EPROM.

InterSystems Standard EPROM Configuration

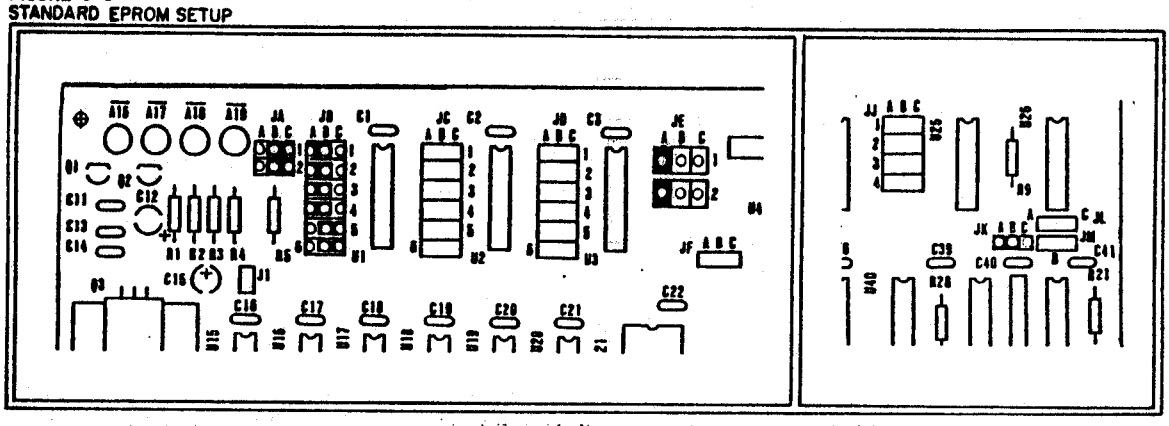
The standard configuration for the jumpers associated with the EPROM are shown in Figure 3-3. This configuration of jumpers represents the following options:

* 2708 EPROM type.
* EPROM address is F000 hex, location of InterSystems bootstrap.
* EPROM disabled.
* Reset jump disabled.
* Single wait state added to EPROM cycles.

To use the EPROM, simply insert your F000 InterSystems Boot EPROM at position U 21, and enable the EPROM by connecting JJ-1 from A to B. A

single wait state will be added to EPROM cycles once the EPROM is enabled. (The EPROM wait state should be disabled for 2 MHZ operation).

FIGURE 3-3
STANDARD EPROM SETUP



If you wish to use the Auto-Boot feature, whereby a jump to the Bootstrap program is executed upon Reset, connect jumper JK from B to C.

3.3 Input/Output Port Setup

Four I/O ports are used on the Series II Z-80 board; they may be addressed at any four-port boundary within the 256 possible system I/O ports. These I/O ports are used to set up and control the Vectored Interrupt Controller and to load the relocation registers of the memory management system.

The base address of the on-board I/O ports, called BASE, is selected with pin-jumpers JD-1 through JD-6, located between U 2 and U 3. The on-board I/O ports are assigned the following functions:

Table 3-3 I/O Port Functions

Port	Function
IN BASE	Read VI Controller Data Port (reads Pre-selected VI register)
OUT BASE	Write to VI Controller Data Port
IN BASE+1	Read VI Controller Status Port
OUT BASE+1	Write to VI Controller Control Port
IN BASE+2	(not assigned)
OUT BASE+2	Write to Page 0 Relocation Register
IN BASE+3	(not assigned)
OUT BASE+3	Write to Page 1 Relocation Register

I/O Base Address Selection

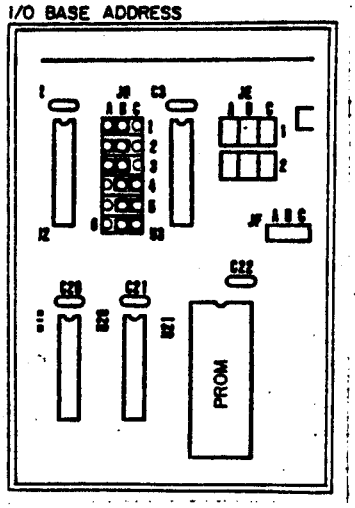
The I/O base address is selected by defining a matching address for the most significant 6 bits of the processors' I/O address. The correspondence between the various settings of the JD jumpers and the processor I/O address are given in the following table. The "Std." column gives the settings for the InterSystems' standard I/O base address, E0 hex.

I/O Port Selection, JD

Jumper	Address Bit	"Zero"	"One"	Std. (E0)
JD-1	A 7	BC	AB	AB
JD-2	A 6	BC	AB	AB
JD-3	A 5	BC	AB	AB
JD-4	A 4	BC	AB	BC
JD-5	A 3	BC	AB	BC
JD-6	A 2	BC	AB	BC

Figure 3-4 illustrates the jumper settings for the standard I/O base address of E0. InterSystems software will use this configuration, and we strongly recommend that unless a specific conflict develops, you use the standard configuration, as software exchange will be greatly facilitated.

FIGURE 3-4
I/O BASE ADDRESS



Port assignments for the standard configuration are:

E0 = VI Data Port
E1 = VI Control/Status Port
E2 = Page 0 Relocation Register
E3 = Page 1 Relocation Register

3.4 Vectored Interrupt Controller Setup

Once the I/O base address has been selected, the Vectored Interrupt Controller may be enabled. The enable for the controller is jumper JJ-2, located between U 24 and U 25 on the right side of the board.

VI Controller Master Enable

JJ-2	A to B	VI Enabled
JJ-2	B to C	VI Disabled

Using an Off-Board Interrupt Controller

The board may be configured such that the on-board interrupt controller is bypassed and the Z-80 processor responds to the group interrupt and issues interrupt acknowledge cycles to the S-100 bus. The vectored interrupt controller must be removed from the board, and jumpers JF and JI should be adjusted. (Caution: the 9519 Controller IC is a static-sensitive chip, and should be stored in anti-static foam, or wrapped in aluminum foil.) Jumper JF is in the center of the board next to the Z-80 chip, and JI is on the right side of the board, between U 11 and U 23.

Bus Interrupt Enables

Function	Jumper	Local Controller	Conventional Interrupts
Bus Int. Accept	JF	B to C	A to B
Bus Int. Assert	JI	B to C	A to B

If these jumpers are not installed, or if they are connected from B to C, they enable the local vectored-interrupt controller circuitry.

3.5 Memory Management Setup

A number of options exist in the setup of the address translator: (1) Either a single 4 K relocation page or 2 independant 4 K relocation pages may be selected. (2) The relocation "Window" may be located anywhere in main memory. (3) The entire address translator may be enabled or disabled with a master enable jumper.

The following table gives the settings for the number of pages and the master enable jumper. The InterSystems standard configuration is to enable both 4 K pages.

Address Translator Size and Enable

Function	JC-4	JC-5	JC-6
Single 4K page	*	AB	BC
Two 4K pages	BC	BC	BC
Translator Disabled	X	X	AB

Where "*" indicates that this jumper participates in the address decoding, and "X" indicates a "don't care" setting.

Locating the Address Translator in Main Memory

Once the size of the translation window has been selected, it may be located in main memory on any 4 K boundary if the window size is 4 K, or on any 8 K boundary if both 4 K pages have been enabled. If the translator has been disabled with the master enable jumper, JC-6, the setting of these address jumpers is insignificant.

The translator is addressed by defining a match address for the first byte of the window area with JC-1 through JC-4. The following table gives the

correspondence between address bits and the jumper settings, as well as the InterSystems standard location for the address translation window, A000 hex.

Address Translator Addressing

Jumper	Address Bit	"One"	"Zero"	Std.
JC-1	A 15	AB	BC	AB
JC-2	A 14	AB	BC	BC
JC-3	A 13	AB	BC	AB
JC-4*	A 12	AB	BC	BC

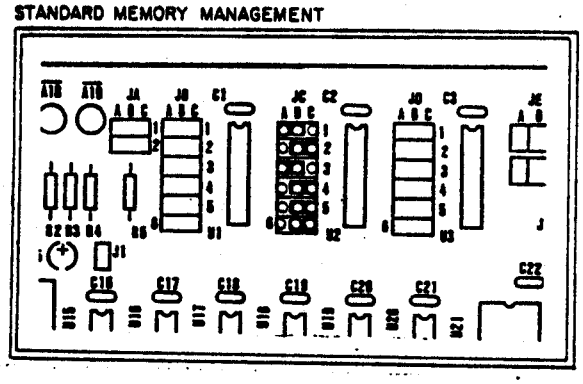
* NOTE: The JC-4 jumper is only used in the address decoding if a single 4 K page has been enabled. If both translation pages are enabled, JC-4 is always connected B to C.

InterSystems Standard Memory Management Configuration

The board is shipped from the factory with the following options selected:

* Base Address is A000 hex.
* 2 pages selected, but the unit is disabled.

FIGURE 3-5
STANDARD MEMORY MANAGEMENT



To use the unit in the standard configuration, simply enable the unit by connecting JC-6 from B to C. The enabled configuration is shown in Figure 3-5. Memory from A000 to AFFF will be relocated as Page 0, and memory from B000 to BFFF will be relocated as Page 1. Note that once the memory management unit is enabled, it must be initialized for proper operation, inasmuch as it will come up at reset with both registers referencing 00,0000H -- the bottom 4 K of main memory.

3.6 Front-Panelless Operation

The Series II Board supports front-panelless operation with a number of jumper selectable features. These include:

* MWRT Memory Write Strobe Generation for all bus devices.

* Front-Panel Control Strobes, Data In Disable and Sense Switch Disable disconnect from the bus.

* Reset, Jump Enable, and Ground available at a 20 pin DIP header for simple control connections.

Memory Write Strobe Generation

The generation of the Memory Write Strobe, MWRT, has traditionally been the function of front panel devices in S-100 systems, where it is implemented as processor memory write OR front-panel deposit.

In a front-panelless system the generation of the memory write strobe becomes the responsibility of the processor. The processor must generate the MWRT signal not only for its own cycles, but also for any Direct Memory Access Device that may have temporary control of the system bus. A circuit to perform this function is included on the Series II processor board, and may be enabled for use in front-panelless systems with pin-jumper JO. MWRT strobe generation must be disabled in front panel systems to prevent conflict with the front panels' generation of the strobe.

MWRT Generation, JO	
MWRT enabled	A to B
MWRT disabled	B to C

The Series II board is shipped with the MWRT strobe disabled.

Front-Panel Control Strobes

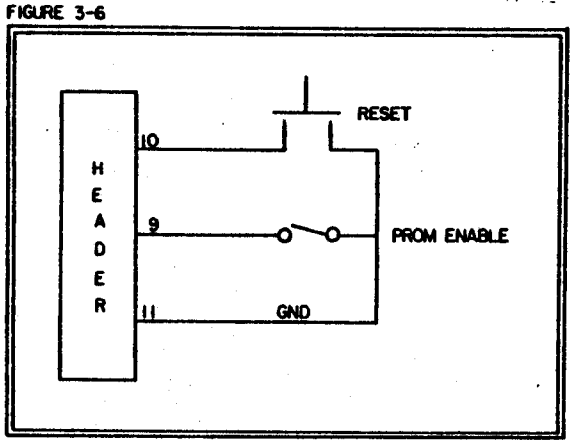
The front panel data control strobes should be disconnected when the board is used in front-panelless systems to avoid possible conflicts and noise. These lines have been included in the bus interface to insure retro-compatibility with pre-IEEE S-100 systems, and may be disabled with jumper JJ-3. The lines which are disabled with this jumper are the Data In Disable signal on bus line 21, and the Sense Switch Disable Signal on bus line 53. Neither of these lines is included in the IEEE S-100 specification.

Front Panel Control Strobes, JJ-3	
Bus strobes disabled	A to B
Bus strobes enabled	B to C

Use of the Front Panel Header in Front Panelless Systems

The 20 pin DIP header which is used to impose data and instructions from the front panel onto the processor's data bus has been designed to ease the connection of basic machine control switches in front panelless systems. The processor RESET line is available at pin 10 of the header, the Reset Jump Enable line is available at pin 9, and a GROUND line is available at pin 11.

FIGURE 3-6



The Connection of these lines is shown in Figure 3-6. When the Boot switch is closed and the Reset pushbutton is depressed, the processor will jump to the on-board PROM if it is enabled, or to the location in main memory set at the PROM address jumpers if the on-board PROM is disabled. If the Boot switch is open and the Reset button depressed, the processor will reset to location 0000.

3.7 Summary of the Standard Setup

As the board is shipped from the factory the following options are set on the Z-80 board:

Standard Setup

Function	Jumpers
BUS INTERFACE:	
4 MHz operation	JG = AB
Full latch mode	JH-1 = BC
M1 wait	JH-2 = AB
No other mem. or IO waits	JH-3, JH-4 = BC
EPROM wait state	JH-5 = AB
NMI to pin 12	JN = AB
Front panel operation	JJ-3 = BC
No MWRT genenration	JO = BC
EPROM:	
2708 EPROM type	JE-1, JE-2 = A
1 K EPROM size	JA-1, JA-2 = AB
EPROM address at F000	JB-1, JB-2, JB-3, JB-4 = AB JB-5, JB-6 = BC
EPROM disabled	JJ-1 = BC
Reset Jump disabled	JK = AB
MEMORY MANAGEMENT:	
Both 4 K pages selected,	JC-1, JC-3 = AB
Window address at A000	JC-2, JC-4, JC-5 = BC
Unit disabled	JC-6 = AB
I/O PORTS:	
I/O address at E0	JD-1, JD-2, JD-3 = AB JD-4, JD-5, JD-6 = BC
VI CONTROLLER:	
On-board VI controller	JI, JF = BC
VI controller enabled	JJ-2 = AB
OTHER:	
No test modes selected	JJ-4, JL, JM = AB

The Standard setup described above is illustrated in Figure 3-7.

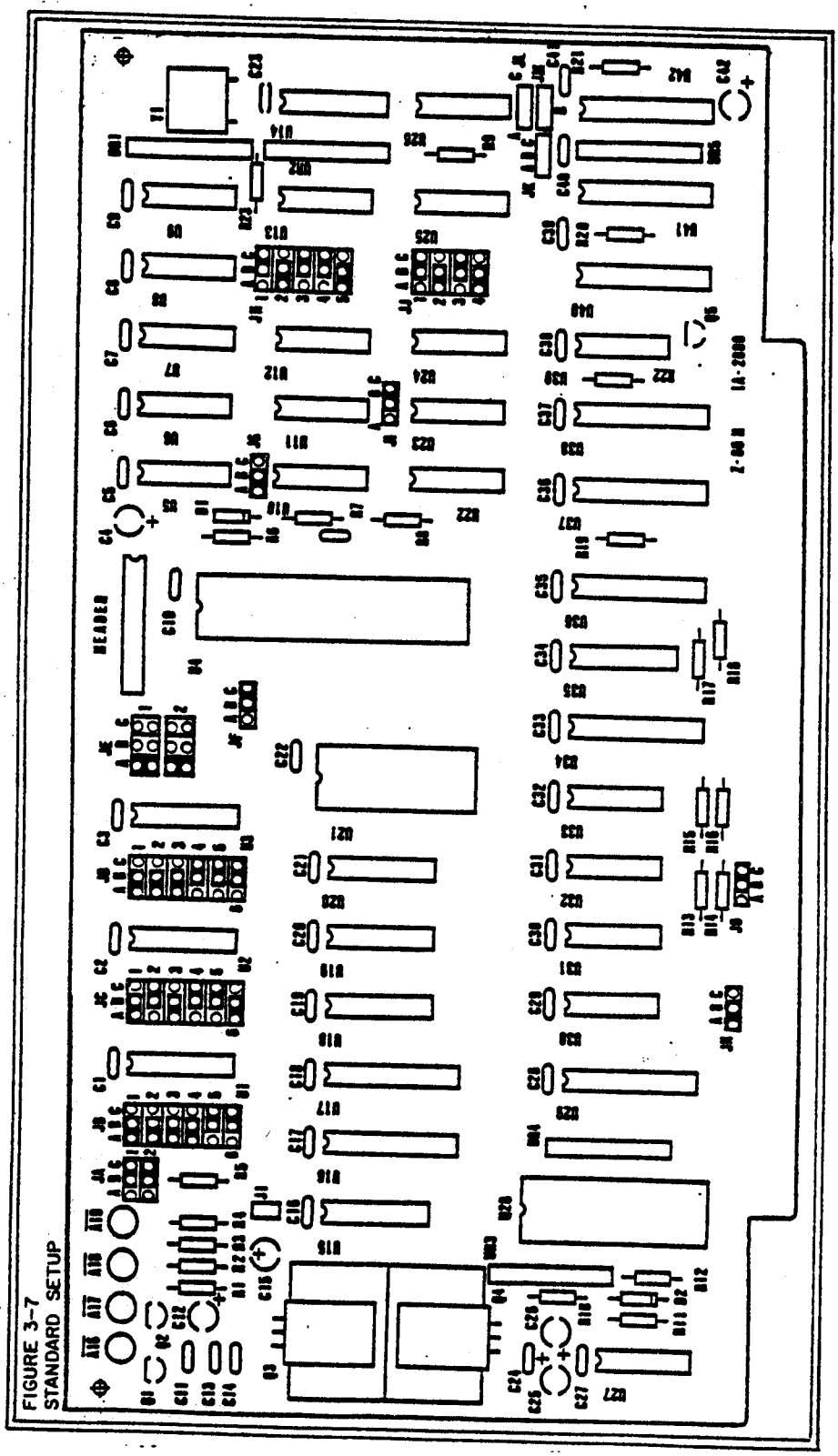


FIGURE 3-7
STANDARD SETUP

3.8 Jumper Summary

Processor Speed, JG	
2 MHz.	B to C
4 MHz.	A to B

Latch Mode, JH-1	
Full latch	B to C
Partial latch	A to B

Wait State Selection, JH-2 thru JH-5

Cycle Type	Jumper	1 wait	no-wait
M1	JH-2	A to B	B to C
All Memory	JH-3	A to B	B to C
All I/O	JH-4	A to B	B to C
On-board ROM	JH-5	A to B	B to C

Non-Maskable Interrupt, JN

NMI to NMI	A to B
NMI to Power Fail	B to C
No NMI to Bus	OPEN

EPROM Type Select Jumpers

TYPE	JA-1	JA-2	JB-5	JB-6	JE-1	JE-2
2708	AB	AB	*	*	A	A
2758	AB	AB	*	*	B	B
2716	AB	BC	*	BC	C	B
2732	BC	BC	BC	BC	C	C
	(NO EPROM, JUMP TO ANY 1 K BOUNDARY)					
NONE	AB	AB	*	*	C	C
	(STANDARD= 2708 AT F000 HEX)					
STANDARD	AB	AB	BC	BC	A	A

* used in addressing

EPROM and Jump Addressing

Jumper	Address Bit	"One"	"Zero"	Std.
JB-1	A 15	AB	BC	AB
JB-2	A 14	AB	BC	AB
JB-3	A 13	AB	BC	AB
JB-4	A 12	AB	BC	AB
JB-5*	A 11	AB	BC	BC
JB-6**	A 10	AB	BC	BC

*Not used in 2732 addressing
**Not used in 2732 or 2716 addressing

EPROM and Jump Enables

Function	JJ-1	JK
No EPROM, No Jump	BC	AB
EPROM enabled, No Jump	AB	AB
Auto-Boot (EPROM and Jump enabled)	AB	BC
Jump to Off-board memory	BC	BC

I/O Port Selection, JD

Jumper	Address Bit	"Zero"	"One"	Std.
JD-1	A 7	BC	AB	AB
JD-2	A 6	BC	AB	AB
JD-3	A 5	BC	AB	AB
JD-4	A 4	BC	AB	BC
JD-5	A 3	BC	AB	BC
JD-6	A 2	BC	AB	BC

VI Controller Enable

JJ-2	A to B	VI Enabled
JJ-2	B to C	VI Disabled

Bus Interrupt Enables

Function	Jumper	Local Cont.	Int. via Bus
Bus Int. Accept	JF	B to C	A to B
Bus Int. Assert	JI	B to C	A to B

Address Translator Size and Enable

Function	JC-4	JC-5	JC-6
Single 4K page	*	AB	BC
Two 4K pages	BC	BC	BC
Translator Disabled	X	X	AB

* Used in addressing
X = Don't Care

Address Translator Addressing

Jumper	Address Bit	"One"	"Zero"	Std.
JC-1	A 15	AB	BC	AB
JC-2	A 14	AB	BC	BC
JC-3	A 13	AB	BC	AB
JC-4*	A 12	AB	BC	BC

MWRT Generation, JO

MWRT enabled	A to B
MWRT disabled	B to C

Front Panel Control Strobes, JJ-3

Bus strobes disabled	A to B
Bus strobes enabled	B to C

3.9 Installing the Board

Once you have selected the operating options for your processor board, you may install the board in your system. We recommend that, for initial check out, you disable the Address Translator by setting JC-6 to AB. If your system depends on bus interrupts using the S-100 signals pINT and INTA, you should consult section 3.4 of this manual regarding disabling the on-board vectored interrupt controller.

With the system power off, insert the board into the S-100 bus, taking care not to skew the card fingers with respect to the edge connector (it is possible with some card edge connector sockets to skew the board such that the gold fingers on the board seat between the brushes of the socket).

Your processor board has been extensively tested before shipment from the InterSystems plant, and should work the first time. If you nevertheless experience difficulties, here are some pointers:

Power down the system. DON'T remove any card immediately; wait a minute for the power supply capacitors to discharge. Check the following:

1) Is the board seated in the edge connector properly?

2) Has one of the chips on the board been jarred loose in shipping?

3) How's your power supply? If the "8 volt" bus is less than 7 volts, this could be the problem.

4) Are the jumpers set properly?

If none of the above solve your problem, or if you need any assistance whatsoever, give us a call at (607) 257-0190.

Section 4.0

Technical Reference Section

4.1 S-100 Not-to-be-Defined Lines

4.2 Test Modes

4.3 Board Timing Diagrams

4.0 Technical Reference

This section of the user's manual describes our use of the "Not-to-be-Defined Lines" on the S-100 bus, overall board timing relationships, and the basic board testing modes.

4.1 S-100 Not-to-be-Defined Lines (NDEFs)

The S-100 bus specification includes 3 lines which are called Not-to-be-Defined Lines (NDEFs). Each manufacturer of products for the bus may use these lines, but must assure that they are not necessary for the proper operation of the product in a general S-100 system. As a general policy, we at InterSystems will not use the NDEF lines for any "new" functions, but will include these lines as options on products where it is felt they provide increased compatibility with pre-standard products.

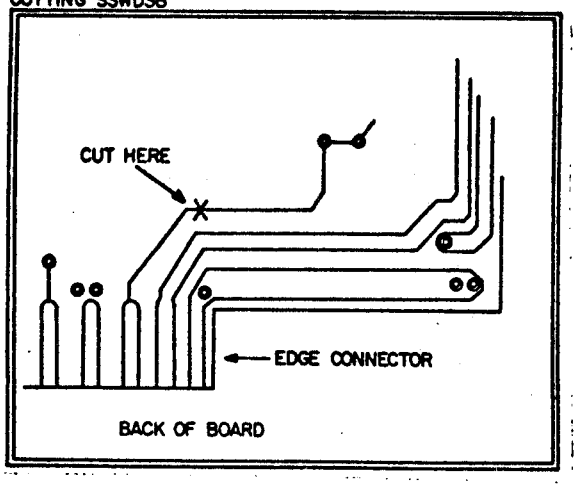
On the Series II Z-80 board, we have used the NDEF line at Pin 21. On traditional front panel devices, pin 21 was used as a data in disable signal, allowing the front panel to impose data and instructions on the processor's data bus. We include the DIDSB signal on the 20 pin DIP front panel connector, but it is also available on the bus, depending on the setting of the jumper JJ-3. Connecting JJ-3 from A to B will enable the DIDSB signal from the 20 pin connector, B to C will enable the DIDSB signal from NDEF line pin 21.

In current InterSystems' DPS-1 computer systems, the DIDSB line is connected between the front panel and the processor via the bus line 21, and a 16-pin cable connects to the front panel header. This implementation will be phased out as mass-terminated 20-pin DIP cables become available in production quantities.

One other line has been included in the processor/bus interface for the sake of retro-compatibility. This line, formerly called Sense Switch Disable, performs the same function as the Data In Disable line but is necessary to the proper operation of IMSAI and Altair front-panels. The SSWDSB line (Bus Pin 53) has been assigned by the IEEE committee as a GROUND. If you have purchased a DPS-1 mainframe system, all data-in-disable functions will be performed by the DIDSB line -- either at pin 21, or on the 20-pin DIP front panel connector.

If you purchased the card separately, and are operating with IMSAI, Altair, or some other front panels, you should probably leave the line connected, but note that your system does not conform to the entirety of the IEEE specification. If any board is installed in the system which implements the line at pin 53 as a ground, the processor will have its data input bus permanently disabled. The line may be cut as shown in Figure 4-1, but the programmed input port on IMSAI or Altair front panels will not function. In front-panelless systems, both lines 21 and 53 are disabled by connecting jumper JJ-3 from A to B.

FIGURE 4-1
CUTTING SSWDSB



4.2 Testing Modes

The Series II Z-80 has two different test modes which will greatly facilitate board checkout. The first of these forces the processor to execute the NOP (no-operation) instruction continuously, and the second test mode executes the RST (restart) instruction continuously.

The NOP test mode exercises the basic control circuits of the Z-80 processor board such that it is extremely easy to obtain a stable display on an oscilloscope. The proper operation of the PSYNC, STVAL, and DBIN control strobes, the address lines (which will count up in binary) and latch control modes may all be tested by comparing the scope waveforms with the timing diagrams given in Section 4.3 . The continuous NOP mode may also be used to test the operation of the PROM address decoders and the address management address decoders.

The continuous Restart mode, on the other hand, does a single M1 read operation, followed by two write operations to the stack. Using this mode the proper operation of the PWR strobe may be easily checked.

The following table gives the jumper settings for each of the test modes. Once the jumpers are set, hit the reset button and run: the processor will execute the test mode.

Testing Modes

Function	JJ-1	JJ-4	JK	JL	JM
Cont. NOP	BC	BC	BC	BC	BC
Cont. RST	BC	BC	AB	BC	BC
Normal	*	AB	*	AB	AB

* Return ROM enable and jump enable jumpers to previous setting.

The proper operation of the S-100 Status lines is easily verified using an InterSystems Front Panel. Deposit the following program in memory:

```
LOOP:   IN      0FFH    ;INPUT FROM THE SENSE SWITCHES
        OUT     0FFH    ;OUTPUT TO THE PANEL LIGHTS
        JMP     LOOP    ;CONTINUE FOREVER
```

Single stepping through the program will check the operation of the status lights. The following sequence will be observed:

Step	Lights On
0	M1, MEMR, WO
1	MEMR, WO
2	INP, WO
3	M1, MEMR, WO
4	MEMR, WO
5	OUT
6	M1, MEMR, WO
7	MEMR, WO
8	MEMR, WO

The proper operation of the address management unit and the vectored interrupt controller may be checked with the test programs given in section 2.0, "Series II Z-80 Architecture".

4.3 Board Timing Diagrams

The timing diagrams in this section (Figures 4-2, 4-3, and 4-4) give the overall bus timing for the Z-80 board. Separate diagrams are given for the two latch control modes.

FIGURE 4-2
FULL LATCH MODE

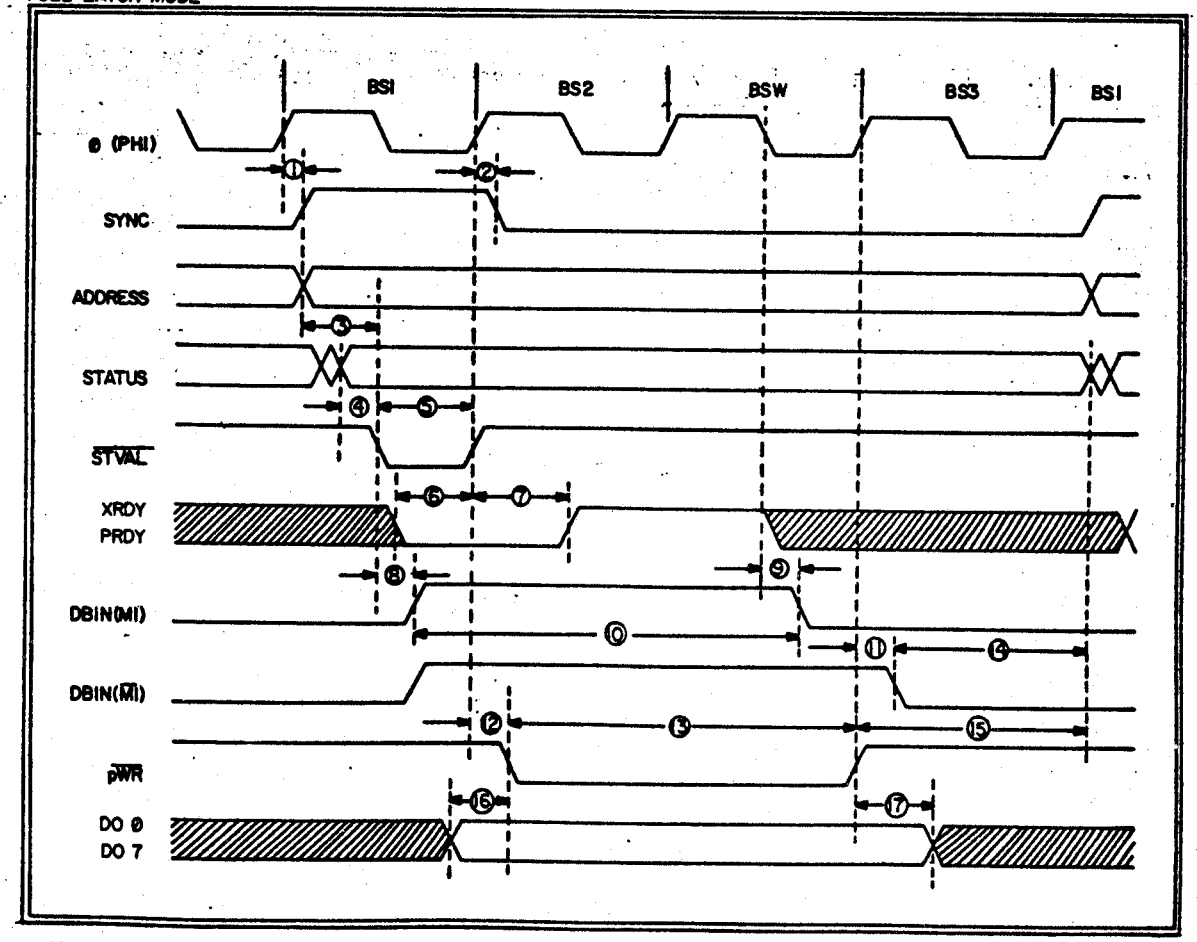


FIGURE 4-3
PARTIAL LATCH MODE

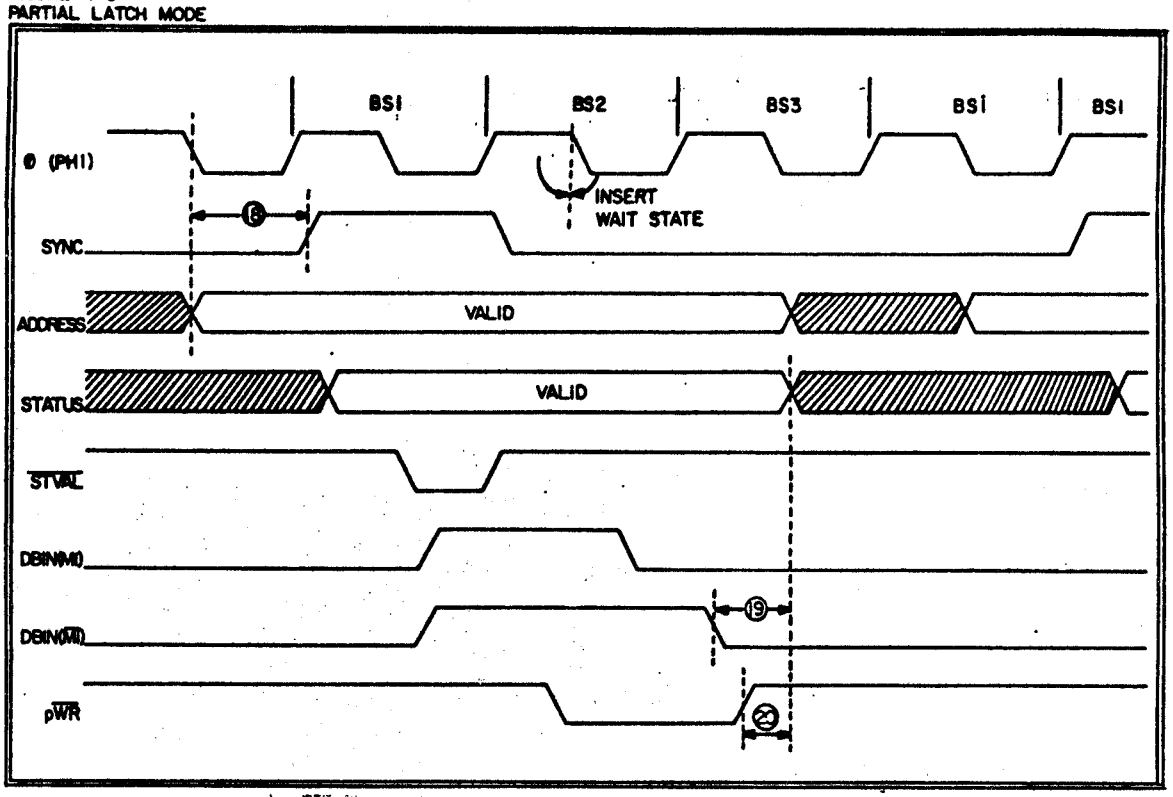
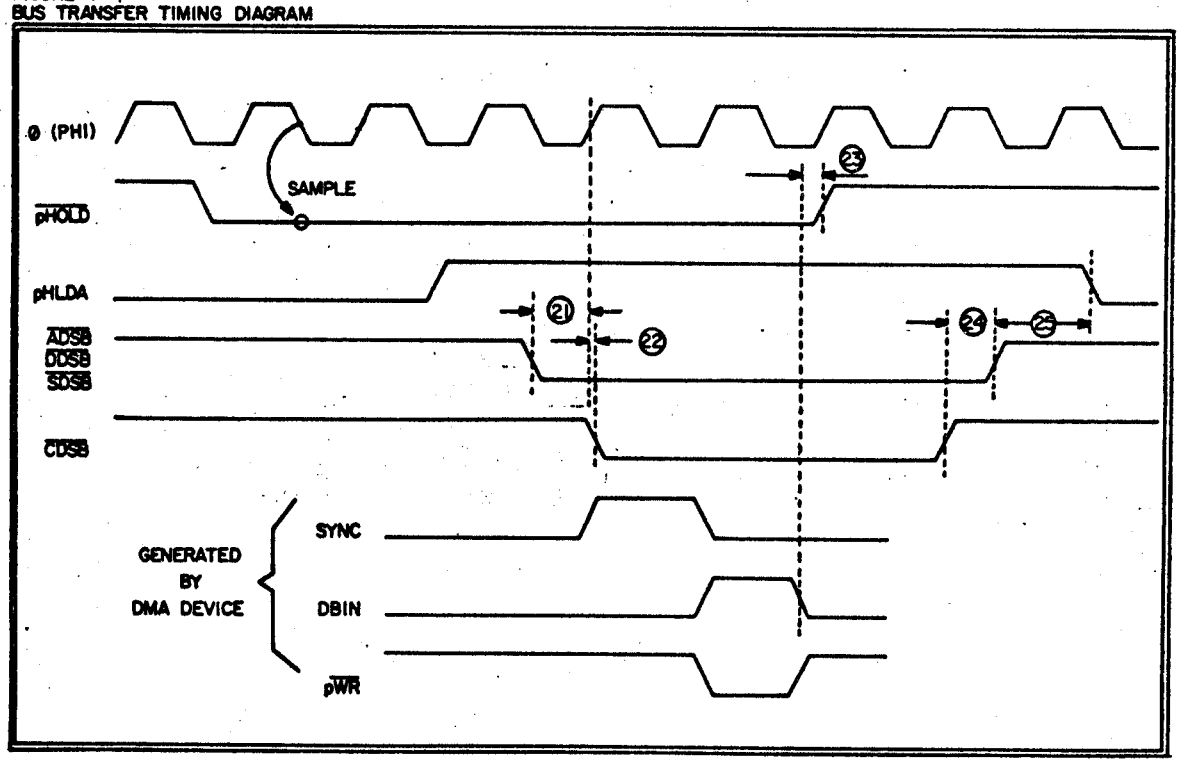


FIGURE 4-4
BUS TRANSFER TIMING DIAGRAM



SERIES II Z-80 TIMING VALUES

#	FUNCTION	MIN	TYP	MAX
		(in nanoseconds)		
1.	Delay: Phi to sync	10	70	100
2.	Delay: Phi to /sync	10	15	20
*3.	Setup: Address to /STVAL, Full Latch	50	70	--
4.	Setup: Status to /STVAL	50	70	--
5.	Duration /STVAL low	100	--	--
6.	Setup: RDY to Phi	75	--	--
7.	Hold RDY from Phi	0	--	--
8.	Delay: /STVAL to DBW	20	--	--
9.	Delay:Phi to /DBIN (M1)	--	--	90
10.	Duration: DBIN High	225	--	--
11.	Delay:Phi to /DBIN (/M1)	--	--	90
12.	Delay:Phi to /PWR	10	--	90
13.	Duration: PWR Low	225	--	--
*14.	Hold STATUS and ADDRESS after /DBIN (Full latch)	295	--	--
*15.	Hold STATUS and ADDRESS after PWR (Full latch)	280	--	--
16.	Setup: DATA OUT to /PWR	100	--	--
17.	Hold DATA OUT from PWR	100	--	--

Partial latch timings differ from those above only in the asterisked cases, as follows:

18.	Setup: ADDRESS to SYNC, (Patial Latch)	80	100	--
19.	Hold ADD & STAT from /DBIN (Partial latch)	50	--	--
20.	Hold ADDRESS & STATUS from PWR (Partial latch)	100	--	--

NOTE: (The expression "/NAME" denotes an active low signal.)

Section 5

Parts List and Parts Placement Diagram

5.0 Parts List and Placement

Figure 5-1 gives the parts placement on the circuit board.

SERIES II Z-80 PARTS LIST

INTEGRATED CIRCUITS POSITION	MANUFACTURERS PART #
U1,U2,U3	8131
U4	Z-80A
U5	74 LS 74
U6	74 LS 04
U7	74 LS 74
U8	74 LS 00
U9	74 LS 74
U10	74 LS 04
U11	74 LS 00
U12	74 LS 04
U13	74 LS 21
U14	74 LS 124
U15	74 LS 157
U16,U17	74 LS 273
U18,U19	74 LS 153
U20	74 LS 157
U21	2708,2758,2716,2732 (NOT INSTALLED)
U22	74 LS 74
U23	74 S 260
U24	74 H 08
U25	74 LS 20
U26	74 LS 74
U27	74 LS 00
U28	9519
U29	74 LS 373
U30	7405
U31	74 LS 02
U32	74 LS 02
U33	74 LS 00
U34	74 LS 373
U35	74 LS 139
U36	74 LS 373
U37	74 LS 244
U38	74 LS 373
U39	74 LS 04
U40,U41,U42	74 LS 244

RESISTORS:

POSITION	VALUE	TOLERANCE	POWER
R1, R2, R3, R4, R7:	330 OHM	10 %	1/4 W
R5, R6, R8, R9, R11, R13, R14, R19, R22:	1 KOHM	10 %	1/4 W
R10, R12, R20:	4.7 KOHM	10 %	1/4 W
R15, R16, R17, R18:	10 OHM	10 %	1/4 W
R21:	47 KOHM	10 %	1/4 W
R23:	130 OHM	10 %	1/4 W

UR1, UR5: 4.7 KOHM 9 PIN SIP (ONE COMMOM) RESISTOR PAK

UR2, UR3, UR4: 1 KOHM 9 PIN SIP (ONE COMMON) RESISTOR PAK

CAPACITORS:

POSITION	VALUE	TYPE	RATING
C4:	1 UF*	DIPPED TANT.	10 V*
C11:	3.3 UF*	DIPPED TANT.	10 V*
C12, C25:	10 UF*	DIPPED TANT.	25 V*
C15, C27:	10 UF*	DIPPED TANT.	10 V*
C26:	10 UF*	DIPPED TANT.	15 V*
C42:	2.2 UF=	DIPPED TANT.	10 V*
ALL OTHERS:	.1 UF	BYPASS	20 V*

*Asterisked capacitor values may actually be greater than those indicated; the "=" sign indicates an exact specification.

OTHER:

POSITION	TYPE	FUNCTION, SPECIFICATION
Q1:	78 L 12	12 V LOW POWER REGULATOR
Q2:	79 L 05	-5 V LOW POWER REGULATOR
Q3, Q4:	7805	+5 V REGULATOR
Q5:	2N 3904	NPN TRANSISTOR
D1, D2:	1N 4148	SIGNAL DIODE
Y1:	4.0 MHZ XTAL	FUNDAMENTAL, 200 OHM
A16, A17, A18, A19:	RED LIGHT EMITTING DIODES	
HEADER:	20 PIN DIP SOCKET	

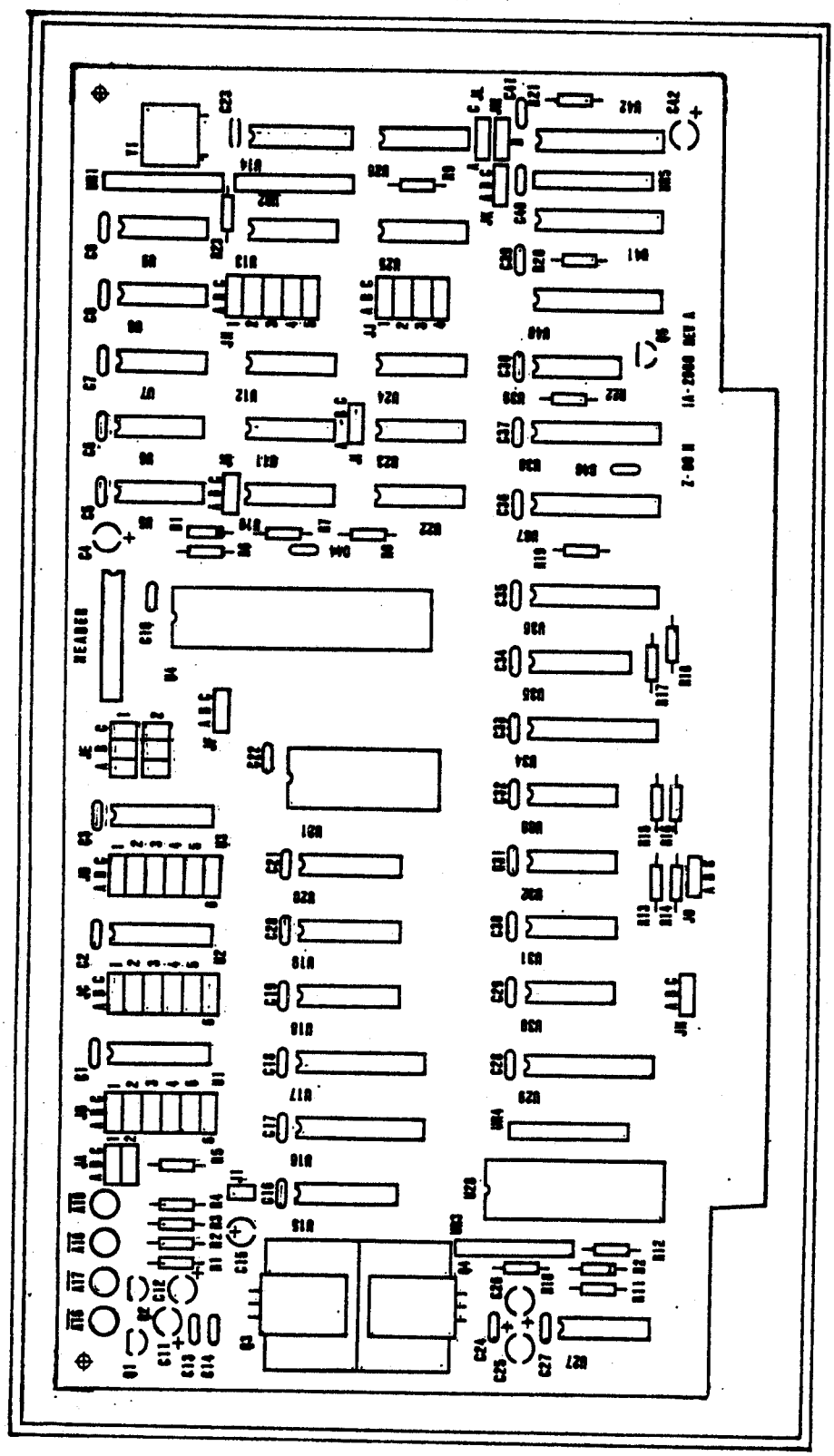


FIGURE 5-1

Section 6

Revisions and Manual Applicability

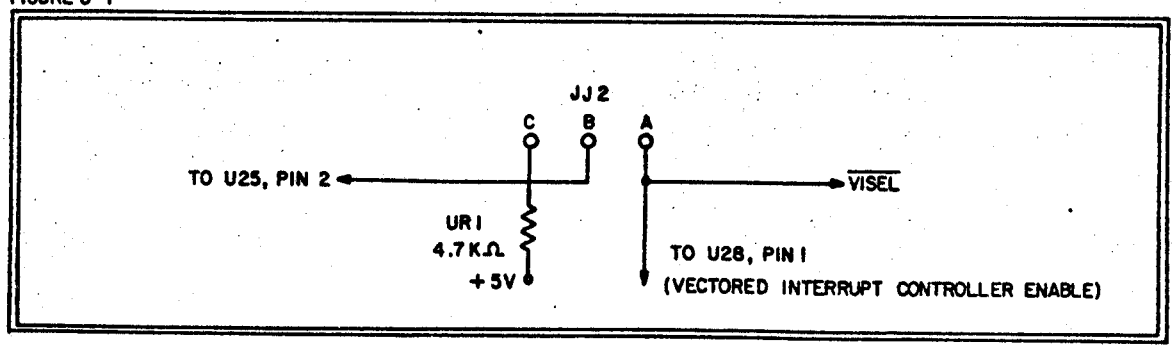
6.0 Revisions and Manual Applicability

This is Revision 0.

A printed circuit error occurs on this revision at jumper JJ-2. The schematic shows the correct configuration; Figure 6.1 below is the actual configuration on this revision. The effect of the error is to enable the vectored interrupt controller according to the I/O jumpers JD-1 through JD-6, whether or not JJ-2 is set for A to B or B to C. The vectored interrupt controller should be removed from the board, in any case, if you wish to use bus interrupts; if this is done, and JJ-2 set with B to C, then the printed circuit error will not affect operation of the card in any way. If you are using the vectored interrupt controller -- and jumper JJ-2 is set A to B, of course -- then, again, operation is not affected.

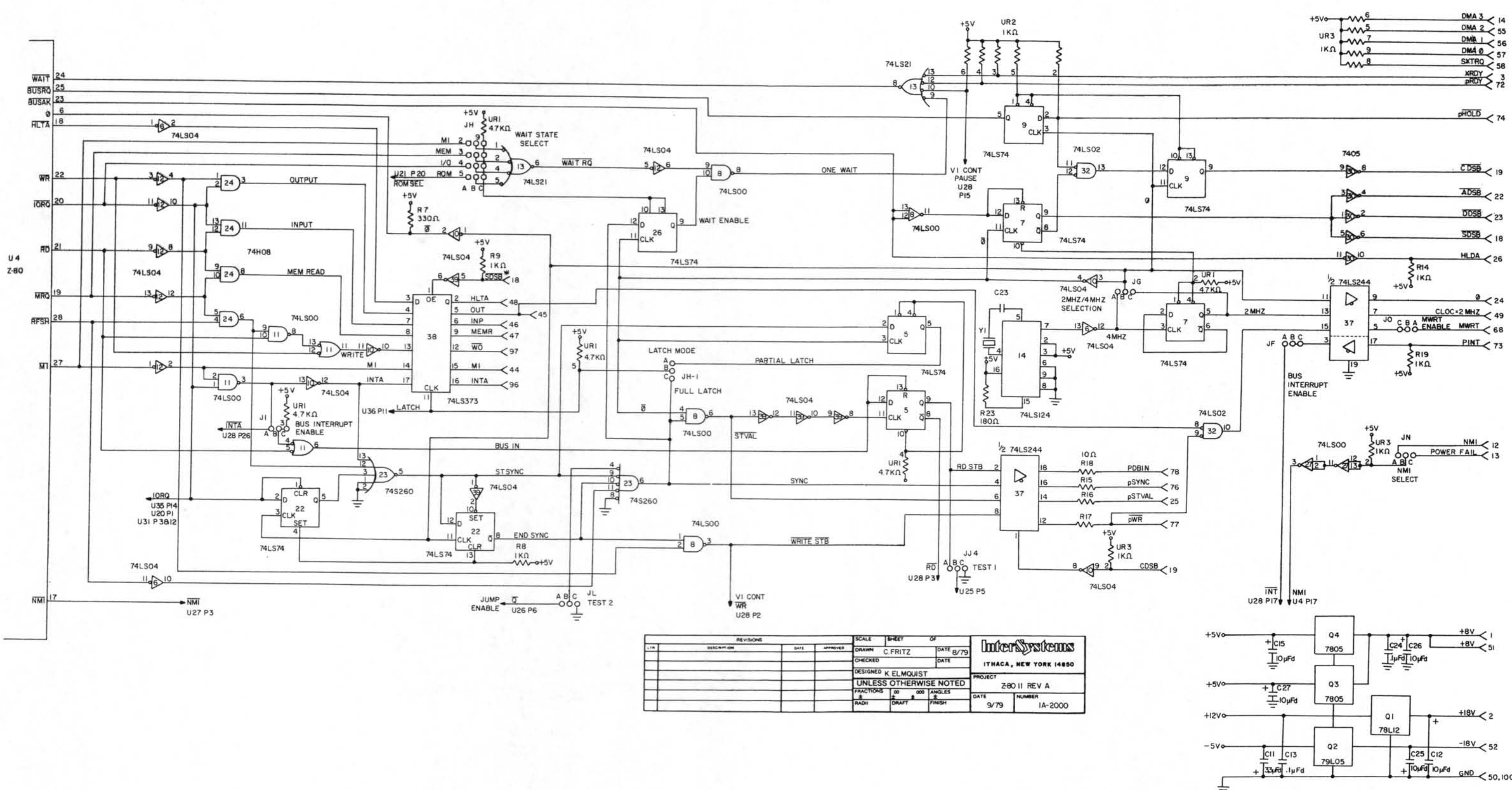
This error will be corrected with Revision A.

FIGURE 6-1



Section 7

Schematic Diagram



U4
Z-80
WAIT
BUSRQ
BUSAK
HLTA
WR
IORQ
RD
MRQ
RFSH
M1
NMI
74LS04
74H08
74LS00
74LS373
74LS74
74S260
74LS21
74LS02
74LS244
74LS124
7405
OUTPUT
INPUT
MEM READ
WRITE
INTA
WAIT STATE SELECT
WAIT RQ
ONE WAIT
WAIT ENABLE
LATCH MODE
PARTIAL LATCH
FULL LATCH
BUS INTERRUPT ENABLE
BUS IN
STSYNC
END SYNC
WRITE STB
SYNC
STVAL
RD STB
2MHZ/4MHZ SELECTION
NMI SELECT
POWER FAIL
JUMP ENABLE
TEST 1
TEST 2
V1 CONT PAUSE U28 P15
DMA 3
DMA 2
DMA 1
DMA 0
SXTRQ
XRDY
pRDY
pHOLD
CDSB
ADSB
DDSB
SDSB
HLDA
CLOC÷2 MHZ
MWRT ENABLE
MWRT
PINT
PDBIN
pSYNC
pSTVAL
pWR
7805
79L05
78L12
InterSystems
ITHACA, NEW YORK 14850
DRAWN C.FRITZ
DATE 8/79
DESIGNED K.ELMQUIST
UNLESS OTHERWISE NOTED
PROJECT Z-80 II REV A
DATE 9/79
NUMBER IA-2000

Z-80
U4

* INDICATES CONNECTION ELSEWHERE ON SCHEMATIC

ERRATA
Edition 1, Z-80 II Manual
May 19, 1980

The following material will be incorporated into Edition 2 of the Z-80 II Manual.

EPROM Jumper Options

The chart at the top of page 46 is in error as regards the settins of the JA-1 and JA-2 jumpers; the same chart is reproduced at the bottom of page 57 and is similarly in error. The standard setup table on page 55 has the same error. The chart on page 46 and page 57 should be as follows:

EPROM Type Select Jumpers

TYPE	JA-1	JA-2	JB-5	JB-6	JE-1	JE-2
2708	BC	BC	*	*	A	A
2758	BC	BC	*	*	B	B
2716	BC	AB	*	BC	C	B
2732	AB	AB	BC	BC	C	C
	(NO EPROM, JUMP TO ANY 1 K BOUNDARY)					
NONE	BC	BC	*	*	C	C
	(STANDARD = 2708 AT F000 HEX)					
STANDARD	BC	BC	BC	BC	A	A

The settings of the JA-1 and JA-2 jumpers in the standard setup table on page 55 (under the "EPROM:" heading) should read:

1 K EPROM size — JA-1, JA-2 = BC

Nomenclature

The Z-80 II board is referred to as the "MPU-80" board in most Intersystems' literature.

Header

The Z-80 II is designed to mate with a front panel via a 16- or 20-pin DIP cable. The 20-pin DIP cable is not available in production quantities at this time, and in most current applications the 16-pin DIP cable is used. The socket provided on the Z-80 II card is, however, a 20-pin socket, providing for upward compatibility when 20-pin DIP cables become available. The 16-pin cable should be used in this socket RIGHT Justified -- that is, holding the MPU-80 card with the S-100 connector pointing down, looking at the component side of the board, the 16-pin cable should go into the header socket so that it lines up with the right side of the header socket, and leaves four pin positions empty at the left side of the header socket.

In the case of the Intersystems front panel, the DIP cable will be properly aligned if no twist is introduced into it -- that is, if the right side of the DIP socket on the front panel is electrically connected to the right side of the DIP socket on the Z-80 II card. As regards other front panels, connect the DIP cable so that data bit zero goes to the right side of the DIP socket on the Z-80 II, data bit 1 goes to the next pin to the left, and so forth.