ALPHA 1

I/O SYSTEM AND INTERFACING GUIDE

```
             ALPHALPHALPHAL
          PHALPHALPHALPHALPHAL
       ALPHALPHALPHALPHALPHALPH
     HALPHALPHALPHALPHALPHALPHAL
    HALPHALPHALPHALPHALPHALPHALP
   HALPHAL        PHALPHALPHALPHA
  HAPHAL           LPHALPHALPHALPH
 HALPH               ALPHALPHALPHA
 ALPH                LPHALPHALPHAL
LPH                 PHALPHALPHALPHA
 HA                 HALPHALPHALPHALP
 AL                HALPHALPHALPHALPH
  AL               ALPHALPH  PHALPHAL
   LP             ALPHALPH   HALPHALP
    HA            LPHALPHA    LPHALPHA
      PH          PHALPHAL    PHALPHAL
        A        PHALPHAL      ALPHALPH
          L     PHALPHALP      LPHALPHA
            P   HAPHALPH        HALPHALP
               HALPHALPH        ALPHALPH
               ALPHALPH          PHALPHAL
               LPHALPHA          HALPHALP
              LPHALPHAL           LPHALPH
              PHALPHAL             PHALPHAL
             PHALPHALP             ALPHALP
             HALPHALP              LPHALPHA
             ALPHALPHA            LPHALPHAL
            ALPHALPHALPHAL      AL  ALPHALP
            LPHALPHALPHALPHA PHA    PHALPHA
           LPHALPHALPHALPHALPH       HALPHALP               A
           PHALPHA       ALPH         LPHALPH              H
          PHALPHALP                    HALPHAL            P
          HALPHALP                     ALPHALP           PH
          ALPHALPH                     LPHALPH         LPH
         ALPHALPH                       HALPHALPHALPHALP
         LPHALPHA                       ALPHALPHALPHALP
        LPHALPHAL                        PHALPHALPHALP
        ALPHALPH                          ALPHALPHALP
       ALPHALPH                            PHALPHALP
```

H E W L E T T - P A C K A R D   P R I V A T E

AUGUST 1971

DO NOT REPRODUCE-----------COPY ____

TABLE OF CONTENTS

LIST OF FIGURES

Figure	Title	Page

SECTION 1

ALPHA I/O OVERVIEW

The design of the Alpha I/O system allows I/O devices to operate independently, and without control, of the CPU. To this end, the I/O bus was given an MCU bus port, shared with the CPU, to ALPHA memory. A simple example of the resultant system is shown in figure 1.

By making the IOP an integral part of the CPU, direct commands can be issued to the I/O devices from the CPU itself. This would be the only mode of operation possible in the system of figure 1.

With the addition of a Multiplexed SIO card or Selector Channel and more sophisticated device controllers, it is possible to control I/O devices without the use of direct CPU commands. Figure 2 details one such possible I/O system. Control of the device controllers in this system is handled by the Multiplexed SIO card or channel. They have the capability of fetching I/O program orders from memory through the IOP and executing them, thereby generating control signals to the device controllers via the SIO bus. The I/O orders and their formats are explained later.

The SIO card can time division multiplex the execution of up to 16 I/O programs. The particular device controller program serviced during any given time slice is determined by a hardware service priority established at system configuration time.

An I/O system incorporating a channel, a multiplexed SIO card, and a simple CPU controller device, is diagrammed in figure 3. Device controller 1 can be operated only under direct CPU control, while device controller 2 can also be controlled by I/O program executed by the multiplexed SIO logic. As with every other device controller, device controller 3 can be directed by CPU commands, in addition to being controlled by the channel.

All Data to the non-channel device controllers and the Multiplexed SIO card is transmitted on the IOP Bus, in addition to the CPU control signals, interrupt system signals, and several SIO-IOP communication signals. A complete IOP bus breakdown is given in figure 4.

The SIO bus provides for individual service request lines for each of the 16 devices that can be operated by the SIO card, 4-coded lines used to inform a particular device controller that it is currently being serviced by the SIO-IOP, and 22 SIO command and device controller response lines. A complete SIO bus breakdown is given in figure 5.

The channel bus replaces the SIO bus on device controllers operating with the channel. The channel bus uses the 16 service request lines for the data path to the device controller, but otherwise creates the same control signals to the device controller as the Multiplexed SIO card.

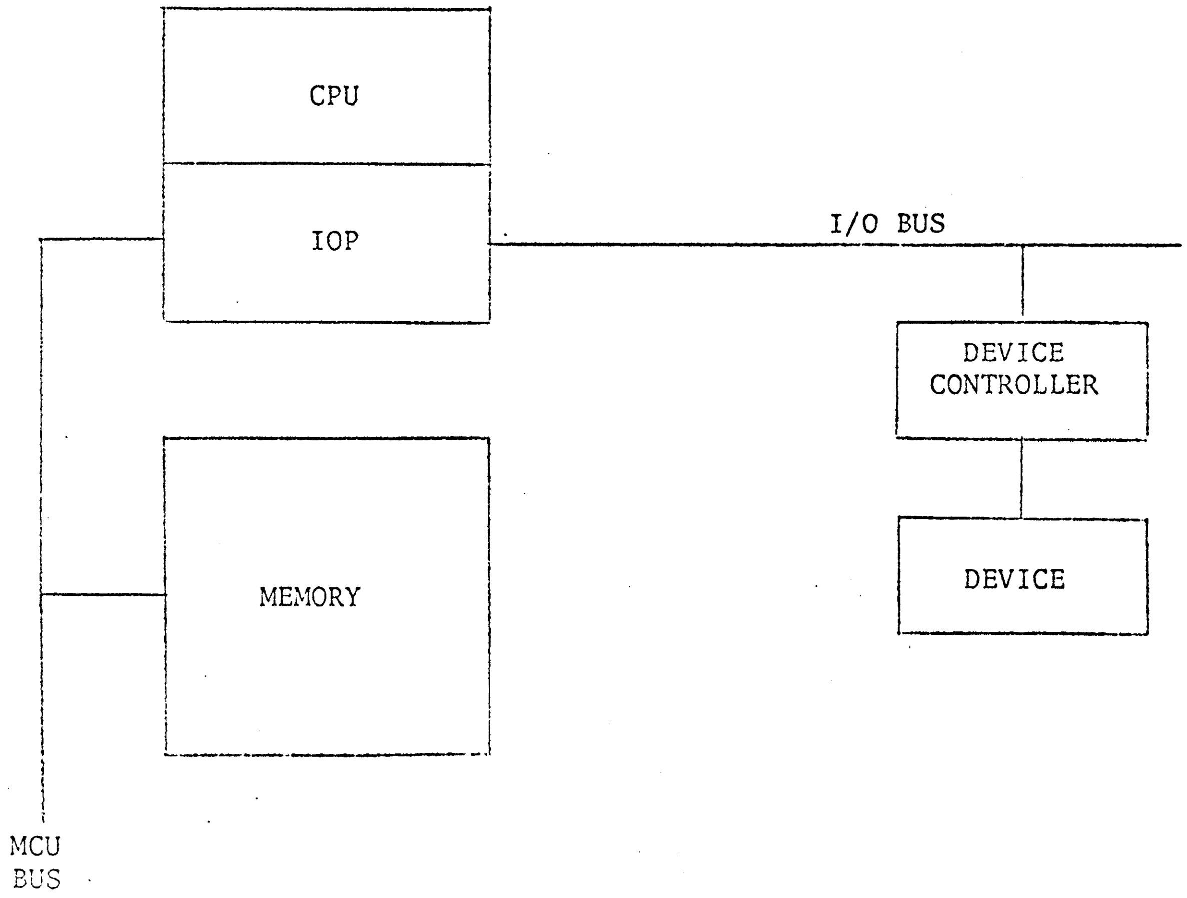


Figure 1. Simple Direct I/O System

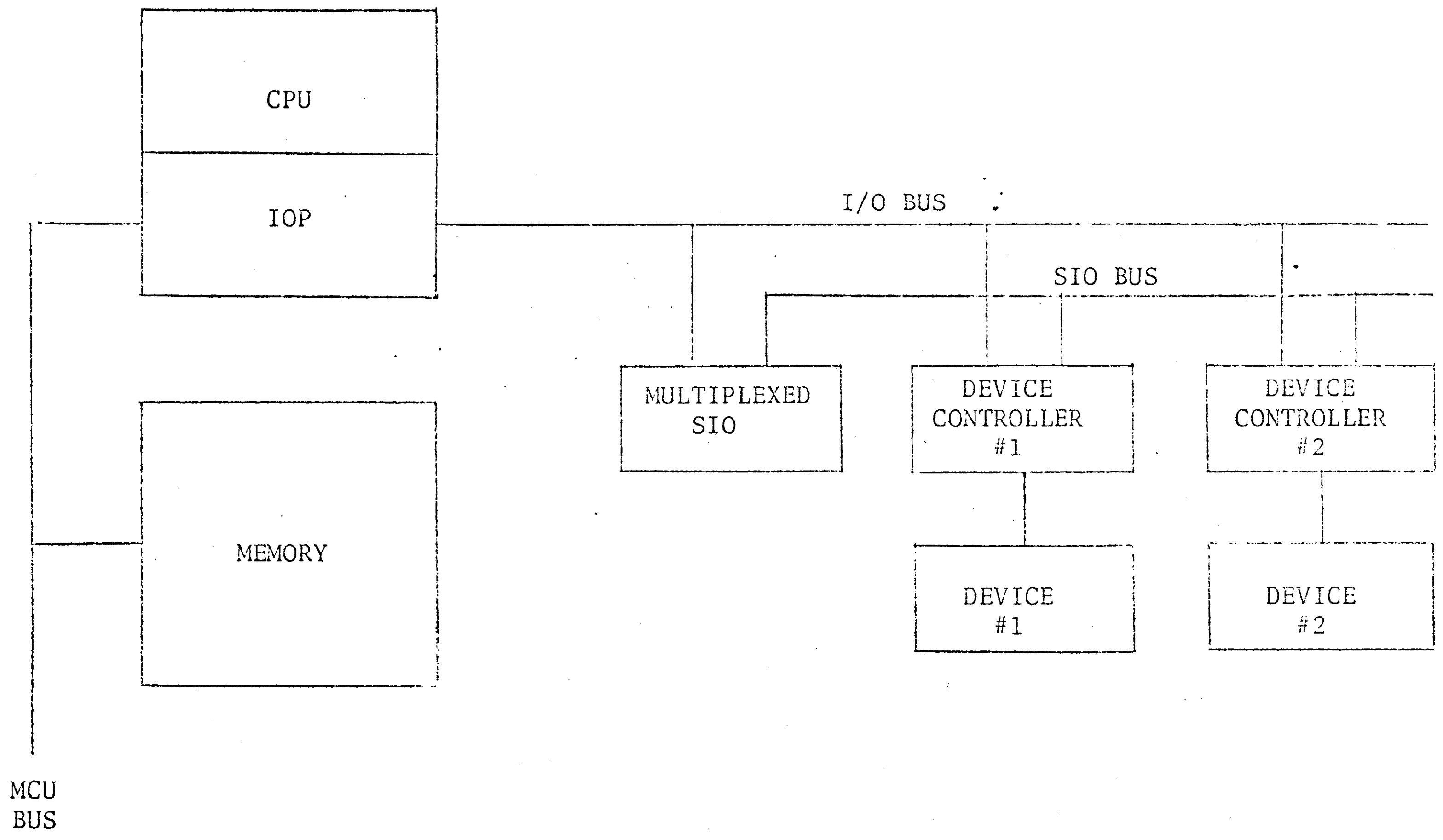


Figure 2. Multiplexed SIO and Direct I/O

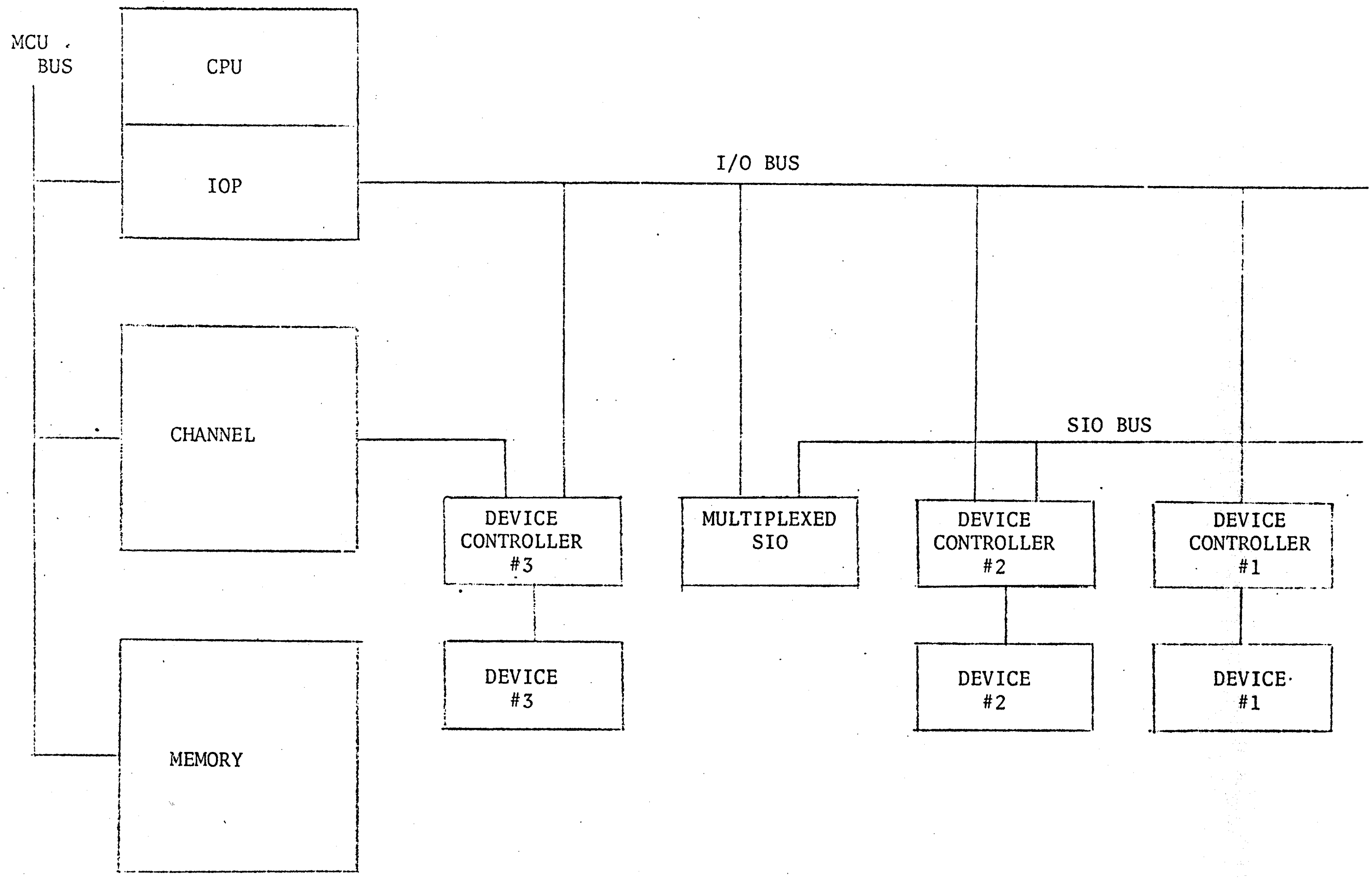


Figure 3. High Speed Channel, Multiplexed SIO, and Direct I/O

1	PARITY LINE	$\overline{\text{PARITY}}$
1	PARITY ERROR LINE	$\overline{\text{XERR}}$
3	DIRECT CPU COMMAND LINES	$\overline{\text{CMD OUT}}$
8	DEVICE ADDRESS LINES	$\overline{\text{DEV ADDR}}$
3	SIO TO IOP COMMAND LINES	$\overline{\text{CMD IN}}$
16	DATA LINES	$\overline{\text{IOP DATA}}$
8	INTERRUPT ADDRESS LINES	$\overline{\text{INT ADDR}}$
4	INTERRUPT REQUEST LINES	$\overline{\text{IREQ}}$
1	INTERRUPT ACKNOWLEDGE LINE	$\overline{\text{IACK}}$
1	SERVICE OUT STROBE LINE	$\overline{\text{SO}}$
1	SERVICE IN RESPONSE LINE	$\overline{\text{SI}}$
1	SERVICE REQUEST LINE	$\overline{\text{SREQ}}$
1	I/O RESET LINE	IORST
1	POWER ON LINE	PON
1	POWER FAIL LINE	PFW
1	CLOCK LINE	CLOCK
1	INBOUND/OUTBOUND	OUTBND
2	DATA POLL LINES (In and Out)	DPOLL*
2	INTERRUPT POLL LINES (In and Out)	IPOLL*

*Separately wired; not part of ribbon cable.

Figure 4. IOP Bus Signals

1	CHANNEL SERVICE OUT LINE	$\overline{\text{CHAN SO}}$
1	CHANNEL ACKNOWLEDGE LINE	$\overline{\text{CHAN ACK}}$
1	SIO INITIATION REQUEST LINE	$\overline{\text{REQ}}$
16	SERVICE REQUEST LINES	$\overline{\text{SRO}}$ thru $\overline{\text{SR15}}$
1	ACKNOWLEDGE SR LINE	$\overline{\text{ACK SR}}$
4	DEVICE SELECT LINES	SELECT
1	ENABLE SIO LINE	$\overline{\text{ENABLE}}$
1	TRANSFER ERROR LINE	$\overline{\text{XFER ERROR}}$
4	TOGGLE LINES	TOGGLE IN/OUT XFER, SR, SIO OK
1	JUMP MET LINE	$\overline{\text{JMP MET}}$
1	DEVICE END LINE	$\overline{\text{DEV END}}$
1	END OF TRANSFER LINE	$\overline{\text{EOT}}$
9	STROBE LINES	$\overline{\text{P STAT STB}}$
		$\overline{\text{SET INT}}$
		$\overline{\text{SET JUMP}}$
		$\overline{\text{RD NEXT WD}}$
		$\overline{\text{P READ STB}}$
		$\overline{\text{P WRITE STB}}$
		$\overline{\text{P CMD 1}}$
		$\overline{\text{P CONT STB}}$
		$\overline{\text{DEV \# DB}}$

Note: High-Speed Channel lines are identical except the 16 Service Request lines are used as high-speed data lines.

Figure 5. Multiplexed SIO Bus Signals

SECTION 2
INPUT/OUTPUT PROGRAMMING

I/O programming is handled by a portion of the CPU microprogram along with special hardware which enables data transfer between devices and memory to proceed in parallel with normal CPU operation.

2.1 CPU INSTRUCTIONS

There are five CPU I/O instructions and addressing capability for 253 device controllers. The I/O instructions are: Start I/O (SIO), Read I/O (RIO), Write I/O (WIO), Test I/O (TIO), and Control I/O (CIO). They expect a device controller number in the stack. For TIO and for a rejected SIO, RIO or WIO, a 16 bit device status word (DSW) is returned to the top of the stack.

Start I/O (SIO) causes the initiation of an I/O program pointed to by an entry in the Device Reference Table (DRT). The DRT is a table beginning in memory location 4 containing at most 253 four word entries. Each table entry corresponds to a unique device number and the first word contains the address of the next I/O command instruction for that device.

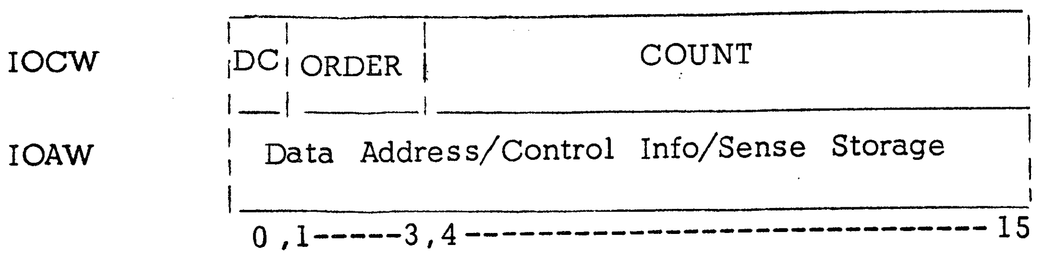
The execution of an SIO causes the transfer of the DRT entry specified by the instruction to be made to the corresponding device controller. The controller then assumes control of the I/O program execution and transfers data to and from the bus. Note that once the device operation has been initiated, the CPU is free to continue processing. Both tasks run concurrently until device termination caused by the appropriate I/O command instruction.

2.2 SIO PROGRAMMING

An SIO type transfer is initiated by the CPU executing a Start I/O instruction for the desired device, assuming that there is an I/O Command program stored in memory.

The DRT entry associated with the device must be pointing to the beginning of the I/O Command program.

The I/O command program consists of a set of doubleword instructions which controls the transfer of data between device and memory. The format of an I/O instruction doubleword is shown in figure 6. Definitions are as follows:



DC	=	Data Chain upon command execution complete. DC bit should be 1 if the ORDER code of the next sequential I/O program doubleword is the same as the current ORDER. This applies only to READ and WRITE ORDER codes.
ORDER	=	I/O operation code
IOCW (4)	=	used on conditional Jump and End.
COUNT	=	Logical transfer count – may be bytes, words or records depending upon the particular device controller. All standard controllers will use a word count. The count is a two's complement negative of the 12 bit count. This field may also be used for control information
IOCW	=	I/O Command Word
IOAW	=	I/O Address Word – the Transfer address or the control information or the storage for the returned status.

Figure 6. I/O Command Doubleword

2.3 I/O COMMAND CODES

CONDITIONAL JUMP (000) If IOCW(4) = 1, a conditional jump to the address given by the IOAW is made at the discretion of the controller. If IOCW(4) = 0, an unconditional jump is made.

RETURN RESIDUE (001) This causes the residue of the count to be returned to the IOAW.

INTERRUPT (010) This causes the device controller to interrupt the CPU.

END (011) End of the I/O program. If IOCW(4) = 1, then the device controller also interrupts the CPU. Device status is returned to the IOAW.

CONTROL (100) This causes transfer of a 16 bit control word in the IOAW to the device controller, as well as the 12 bit count field.

SENSE (101) This causes transfer of a 16 bit status word from the device to the IOAW.

WRITE (110) This causes COUNT words of data to be transferred between core and the device starting at the address given by the IOAW.

READ (111) This causes COUNT words of data to be transferred between the device and core starting at the address given by the IOAW.

2.4 EXTERNAL INTERRUPTS

The external interrupt structure is a "polling" structure with a maximum of 253 devices allowed on the Interrupt Poll (IP) line.

Servicing of the external interrupts is done in descending order of priority, i.e., the highest priority interrupt is serviced first. The interrupt priority of a device is determined by its logical proximity to the CPU on the IP line. The interrupt structure is nested such that a higher priority interrupt can pre-empt a lower one. A 16 bit Mask register is provided for the purpose of masking off groups of external interrupts. Up to 16 external interrupts may be assigned to a mask group.

In the ALPHA I/O system there are four characteristic numbers or values associated with an I/O device. (Note that they are fixed at hardware system configuration time.) These are Device Number, Data Service Priority, Interrupt Priority and Interrupt Mask Number. These characteristic values are all independent of each other, giving the following advantages:

1. Device Numbers may be numbered consecutively, starting at 3 and proceeding to the number of devices on the system. When a new I/O device is added to the system, it is merely assigned the next highest available number, if desired.

2. Since both Data Service and Interrupt Priorities are independent of device number, a new device to the system may be placed anywhere in the priority chain, independent of physical location within the cabinet.

3. Since Data Service Priority and Interrupt Priority are independent of each other, a device which requires a high data transfer rate may be assigned a low interrupt priority, if desired (such as a disc), or a device which has a very low data rate may be configured to a high priority interrupt (such as an alarm condition).

4. Since Interrupt Masks are independent of device numbers and priorities, devices may be masked in groups related to any desired function. For example, if two data terminals were on the system and each has both a high speed and a low speed device, the interrupts could be masked for each terminal, rather than on a data rate basis.

Each device has five interrupt states:

1. Quiescent: Device is not attempting to request an interrupt.

2. Masked: Device is attempting to request an interrupt but its Mask bit is set (=1).

3. Requesting: Device is requesting an interrupt and its Mask bit is clear (=0) but there is a higher priority interrupt current and/or active.

4. Current: Device's interrupt routine is currently being executed on the Interrupt Control Stack (ICS). The device's Active bit is set.

5. Active: Device's Active bit is set, but a higher priority interrupt has occurred which is now executing on the ICS.

The interrupt response time is defined to be the maximum time that may elapse between setting of the active state of an interrupt and the start of the execution of the first instruction of the interrupt routine for the highest priority unmasked interrupt. This response time is approximately 30 microseconds.

SECTION 3

IOP BUS DEFINITION

3.1 INTRODUCTION

IOP bus consists of 52 signals. These signals, by function, can be divided into four groups:

(a) Request-Response group: which consists of the signals that are raised to request transfer of information, or as a response to such request. These signals are $\overline{\text{SREQ}}$, $\overline{\text{SI}}$, $\overline{\text{SO}}$, DPOLL, $\overline{\text{IREQ}}$, $\overline{\text{IPOLL}}$, $\overline{\text{IACK}}$.

(b) Control group: These are the signals that control the type and the direction of the transfer that must take place. Control group consists of $\overline{\text{CMD OUT}}$ bus (0, 1, 2) and $\overline{\text{CMD IN}}$ bus (0, 1, 2).

(c) Information group: These lines carry information between the IOP and the device adaptors. Information group consists of $\overline{\text{IOP DATA}}$ bus (0-15), $\overline{\text{I/O ADDR}}$ bus (0-7), $\overline{\text{INT ADDR}}$ bus (0-7), $\overline{\text{XERR}}$, $\overline{\text{PARITY}}$, INBOUND/OUTBOUND.

(d) System Status group: These lines carry system power status to the I/O adaptors. Status lines consist of PFW, PON, IORST.

The following three sections describe the IOP bus signals.

3.2 IOP BUS, FUNCTIONAL DESCRIPTION

3.2.1 CMD OUT Bus

The $\overline{\text{CMD OUT (0, 1, 2}}$) bus is a unidirectional bus that carries the I/O commands to the I/O adaptors. The I/O commands and their CMD OUT codes are as follows:

IO COMMAND	CMD OUT CODE
SET INT	000
RESET INT	001
START I/O (SIO)	010
SET MASK	011
CONTROL I/O	100
TEST I/O	101
WRITE I/O	110
READ I/O	111

CMD OUT bus must only be interpreted by a device controller if the following two conditions are true:

(a) $\overline{\text{DEV ADDR}}$ bus (0-7) contains the address of IO device.

(b) $\overline{\text{SO}}$ signal is asserted.

$\overline{\text{CMD OUT}}$ bus is activated by the IOP one clock time before the $\overline{\text{SO}}$ signal is asserted. This is to allow the $\overline{\text{CMD OUT}}$ bus to stabilize, at the device adaptor end, before any strobing takes place.

3.2.2 DEV ADDR Bus

This is an outbound bus that carries the device address of the device that must interpret the I/O command, currently contained in the $\overline{\text{CMD OUT}}$ bus. $\overline{\text{DEV ADDR}}$ bus (0-7) is activated by the IOP one clock time before the $\overline{\text{SO}}$ signal is asserted.

3.2.3 IOP DATA Bus

This is a bidirectional bus that carries the I/O data, control, status, and I/O memory address to or from the device adaptors as follows:

(a) During the outbound transfers IOP activates the I/O bus one clock time before the $\overline{\text{SO}}$ signals are asserted. This is to allow the $\overline{\text{IOP DATA}}$ bus to stabilize before any strobing by the device adaptors takes place. $\overline{\text{IOP DATA}}$ bus remains active for at least one clock time after the $\overline{\text{SO}}$ is deactivated.

(b) During the inbound transfers IOP expects to receive $\overline{\text{SI}}$ signal as an indication that the $\overline{\text{IOP DATA}}$ bus contains a message from the device controller. Data must be gated on to the $\overline{\text{IOP DATA}}$ bus by the I/O adaptor no later than the time $\overline{\text{SI}}$ is asserted.

3.2.4 SERVICE OUT ($\overline{\text{SO}}$)

The $\overline{\text{SERVICE OUT}}$ signal is outbound, and is used as follows:

(a) During the execution of direct commands, $\overline{\text{SO}}$ is used to indicate the validity of the $\overline{\text{CMD OUT}}$ bus, $\overline{\text{DEV ADDR}}$ bus, and $\overline{\text{IOP DATA}}$ bus (WIO, CIO only).

(b) During the outbound SIO transfers, SO is used to indicate the validity of the $\overline{\text{IOP DATA}}$ bus.

(c) During the inbound SIO transfers, $\overline{\text{SO}}$ is used as the IOP request for data. Device controller must respond by returning the inbound data and asserting $\overline{\text{SI}}$.

In all cases $\overline{\text{SO}}$ is responded to by $\overline{\text{SI}}$.

3.2.5 SERVICE REQUEST ($\overline{\text{SREQ}}$)

This line is asserted by the device adaptors to request transfer of a word of data to or from memory. This line, once activated by a device adaptor, must be kept active until DPOLL reaches that device adaptor.

3.2.6 DPOLL

This signal is the IOP's response to the SREQ; also it marks the beginning of the transfer cycle. DPOLL line is "daisy chained" among the device adaptors according to their priority. The higher priority devices are closer to the IOP on DPOLL line, thus if two device adaptors request service simultaneously, the higher priority multiplexed SIO card stops the Poll from reaching the lower priority multiplexed SIO card. The adaptor that stops the poll must:

(a) Deactivate its $\overline{\text{SREQ}}$

(b) Assert $\overline{\text{SI}}$ in conjunction with $\overline{\text{CMD IN}}$ code (see section 3.2.7)

(c) Gate the Memory address onto the $\overline{\text{IOP DATA}}$ bus no later than when $\overline{\text{SI}}$ is asserted.

IOP drops DPOLL on the same clock that it loads the Memory Address Buffer from the $\overline{\text{IOP DATA}}$ bus.

3.2.7 CMD IN Bus

This bus is an inbound unidirectional bus that carries the transfer code to the IOP. Transfer code indicates to the IOP the type of transfer that must be performed during the current transfer cycle. Transfer codes are as follows:

TRANSFER CODE 0 1 2	TYPE OF TRANSFER
1 0 0	Chain (DRTE fetch)
0 1 0	JMP
0 0 1	Memory Bound
0 0 0	Device Bound

$\overline{\text{CMD IN}}$ bus will be deactivated once the trailing edge of the DPOLL has reached the multiplexed SIO card.

3.2.8 SERVICE IN ($\overline{\text{SI}}$)

This signal is the device controllers and multiplexed SIO response to either DPOLL or $\overline{\text{SO}}$ signals, as follows:

(a) As a response to DPOLL, $\overline{\text{SI}}$ is asserted by the I/O adaptor to indicate that $\overline{\text{CMD IN}}$ bus contains the transfer command code and

$\overline{\text{IOP DATA}}$ bus contains the memory address of the transfer. The address will be loaded in Memory Address Buffer in the IOP.

(b) As a response to $\overline{\text{SO}}$, $\overline{\text{SI}}$ is asserted by the I/O adaptor either as "data received" signal (during outbound transfers), or as an indicator that $\overline{\text{IOP DATA}}$ bus contains the memory data (during inbound transfers).

For the cases where $\overline{\text{SI}}$ must be used to strobe the I/O bus to strobe the I/O bus to either the IOP Memory Address Register or Memory Data Register, $\overline{\text{SI}}$ is always skewed by the IOP for 1/2 clock time (to allow the bus to stabilize) before it is used to strobe the bus. $\overline{\text{SI}}$ must be deactivated when $\overline{\text{SO}}$ or DPOLL signals are deactivated.

3.2.9 INTERRUPT REQUEST ($\overline{\text{IREQ}}$)

This signal is used by the IO device controller as a request for software program interruption. IOP responds to the $\overline{\text{IREQ}}$ by asserting its IPOLL. The requesting controller must deactivate its $\overline{\text{IREQ}}$ when it has received IPOLL.

3.2.10 INTERRUPT POLL ($\overline{\text{IPOLL}}$)

This signal is initiated by the IOP as the response to $\overline{\text{IREQ}}$. Its function is similar to that of DPOLL line. The requesting device controller that is reached by the IPOLL must respond as follows:

(a) If its $\overline{\text{IREQ}}$ is active, it must stop IPOLL propagation and assert $\overline{\text{IACK}}$. It must also gate its interrupt address onto the $\overline{\text{INT ADDR}}$ bus.

(b) If it is in "interrupt active" state, it must assert its $\overline{\text{IACK}}$ and gate 0's onto the $\overline{\text{INT ADDR}}$ bus.

(c) If it is neither requesting nor active, it must let the IPOLL propagate.

3.2.11 INTERRUPT ACKNOWLEDGE ($\overline{\text{IACK}}$)

This signal is initiated by the device controllers as the response to IPOLL. It is used by the IOP to strobe the interrupt address into the Interrupt Address Register.

3.2.12 INTERRUPT ADDRESS Bus

The $\overline{\text{INT ADDR}}$ bus (0-7) is inbound unidirectional bus that is used by the device controllers to transfer the interrupt address of the currently requesting device to the IOP. IOP uses the skewed (by 1/2 clock time) $\overline{\text{IACK}}$ to strobe the interrupt address into the Interrupt Address Register. If interrupt address is zero when $\overline{\text{IACK}}$ is active, this is an indication that $\overline{\text{IPOLL}}$ was stopped by a device adaptor that has a currently active interrupt which is higher priority than the requesting device.

3.2.13 CLOCK

This line carries the system clock to the device controllers. System clock is a symmetrical square wave.

3.2.14 POWER ON (PON)

This signal is sourced at the CPU power supply. It is the indication of the states of the DC supplies in the system. Its level is above 3.5 volts as long as all the DC supplies in the system remain above their threshold voltage, and drops to ground level when any of the DC supplies reaches below their threshold voltage. (See figure 7.)

3.2.15 POWER FAIL WARNING (PFW)

This line is normally high until 500 μs before any of the DC supplies fail, at which point it drops to ground level. (See figure 7.)

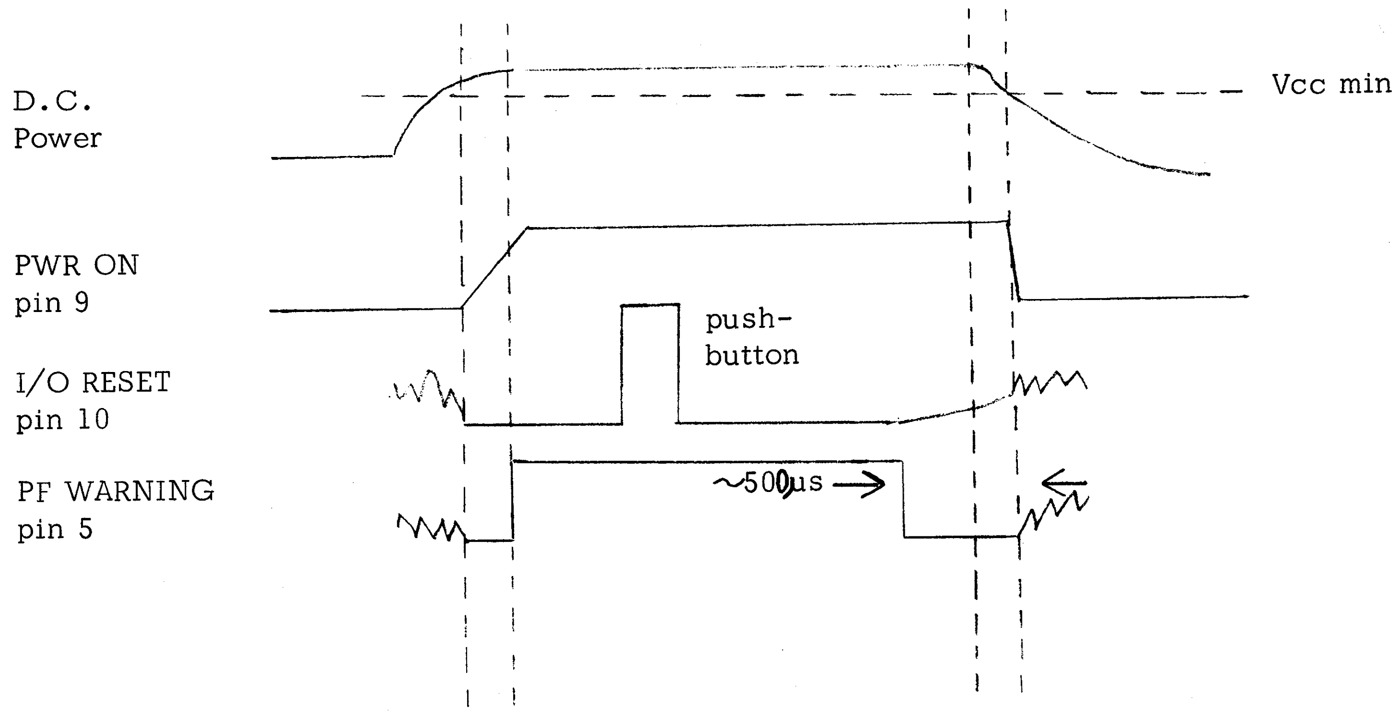


Figure 7. Power Level Timing

3.2.16 I/O RESET

The IORST line is normally low, it goes up to true state when I/O RESET button is depressed. (See figure 7.)

3.2.17 TRANSFER ERROR ($\overline{\text{XERR}}$)

This signal is an outbound signal. It is asserted by the IOP when one of the following error conditions has been detected in the current transfer:

(a) Illegal Address. This error comes up when the IOP receives a memory address that is higher than the highest address in the memory system. IOP cancels the current transfer when this error is detected.

(b) Memory Address Parity Error. This error is detected by the memory.

(c) Memory System Error. This error is detected by the memory. The device controller that is currently engaged in data transfer must strobe $\overline{\text{XERR}}$ with the trailing edge of $\overline{\text{SO}}$ signal.

3.2.18 DATA PARITY

This line carries odd parity for the IOP DATA bus (a) during the SIO address transfer and (b) during the outbound SIO data transfer. Data Parity Line is true when the IOP DATA bus is true.

3.2.19 OUTBOUND SIGNAL

The IOP will bring a data direction signal called OUTBND to P1-12 and P1-11. These pins are spares and will not be carried along the bus. If at some future date it is decided to extend the IOP bus and line drivers/buffers are needed then OUTBND will be necessary to control the Data Bus buffer logic.

3.3 PHYSICAL DESCRIPTION

The IOP bus consists of a 50-wire ribbon cable that occupies the P3 connector position of the I/O backplane, and a portion of the 56-pin connector printed circuit cable that serves also as the power distribution cable. Assignment of signals to the wires is listed in Tables 1 and 2.

3.4 ELECTRICAL DESCRIPTION

I/O bus lines, except for PON, are driven and received by HP 104B and HP 106B respectively. The characteristics impedance of each line is approximately 90 ohms. Each line is terminated at two extreme ends with a terminating network. A discussion of the terminating techniques follows.

Clock signal is available to the device controllers. It must be noted that IOP bus is a true "handshake" bus, and the clock signal is not needed in interfacing to this bus. Device controllers that use the clock for their internal use must be prepared to run at variable clock rate, since the clock cycle varies depending on whether the system is running under internal clock or the external clock mode. Under external clock mode there is no guaranteed minimum clock speed or maximum clock period.

Table 1

IOP BUS PIN ASSIGNMENTS, CONNECTOR P3

IOP PIN NUMBER	I/O SIGNAL NAME	IOP PIN NUMBER	I/O SIGNAL NAME
29	1 SPARE XERR	57	26 $\overline{\text{DATA 7}}$
33	2 SPARE PARITY	56	27 $\overline{\text{DATA 8}}$
31	3 SPARE GND	59	28 $\overline{\text{DATA 9}}$
35	4 $\overline{\text{CMD OUT 0}}$	58	29 $\overline{\text{DATA 10}}$
32	5 $\overline{\text{CMD OUT 1}}$	61	30 $\overline{\text{DATA 11}}$
37	6 $\overline{\text{CMD OUT 2}}$	62	31 $\overline{\text{DATA 12}}$
34	7 $\overline{\text{DEV ADDR 0}}$	63	32 $\overline{\text{DATA 13}}$
41	8 $\overline{\text{DEV ADDR 1}}$	64	33 $\overline{\text{DATA 14}}$
36	9 $\overline{\text{DEV ADDR 2}}$	65	34 $\overline{\text{DATA 15}}$
43	10 $\overline{\text{DEV ADDR 3}}$	--	35 GND
38	11 $\overline{\text{DEV ADDR 4}}$	67	36 $\overline{\text{INT ADDR 0}}$
45	12 $\overline{\text{DEV ADDR 5}}$	66	37 $\overline{\text{INT ADDR 1}}$
42	13 $\overline{\text{DEV ADDR 6}}$	69	38 $\overline{\text{INT ADDR 2}}$
47	14 $\overline{\text{DEV ADDR 7}}$	68	39 $\overline{\text{INT ADDR 3}}$
44	15 $\overline{\text{CMD IN 0}}$	71	40 $\overline{\text{INT ADDR 4}}$
49	16 $\overline{\text{CMD IN 1}}$	70	41 $\overline{\text{INT ADDR 5}}$
46	17 $\overline{\text{CMD IN 2}}$	73	42 $\overline{\text{INT ADDR 6}}$
--	18 GND	72	43 $\overline{\text{INT ADDR 7}}$
48	19 $\overline{\text{DATA 0}}$	--	44 GND
51	20 $\overline{\text{DATA 1}}$	74	45 $\overline{\text{INT REQ 1}}$
50	21 $\overline{\text{DATA 2}}$	75	46 $\overline{\text{INT REQ 2}}$
53	22 $\overline{\text{DATA 3}}$	76	47 $\overline{\text{INT REQ 3}}$
52	23 $\overline{\text{DATA 4}}$	77	48 $\overline{\text{INT REQ 4}}$
55	24 $\overline{\text{DATA 5}}$	--	49 GND
54	25 $\overline{\text{DATA 6}}$	79	50 $\overline{\text{INT ACK}}$

Table 2

56 PIN POWER BUS (PC BACKPLANE)

PIN	IOP PIN	SIGNAL NAME	SIGNAL NAME	IOP PIN	PIN
55		POLL GND	DPOLL IN		56
53	20	$\overline{\text{SO}}$	SO GND	19	54
51		POLL GND	DPOLL OUT		52
49	18	$\overline{\text{SI}}$ SIG	$\overline{\text{SI}}$ GND	17	50
47		POLL GND	INT POLL		48
45	16	$\overline{\text{SREQ}}$	$\overline{\text{SREQ}}$ GND	15	46
43	14	POLL GND	INT POLL OUT	13	44
41					42
39		+20	+20		40
37		+20	+20		38
35		-20	-20		36
33		-20	-20		34
31		-20	-20		32
29		GND	GND		30
27		-15	-15		28
25		-15	-15		26
23		+15	+15		24
21		+15	+15		22
19		GND	GND		20
17		-5	-5		18
15		GND	GND		16
13	8	CLK GND	CLK	7	14
11		SPARE	SPARE		12
9	4	PON	IORST	3	10
7		SPARE	SPARE		8
5	9	PF WARNING	SPARE		6
3		+5	+5		4/2

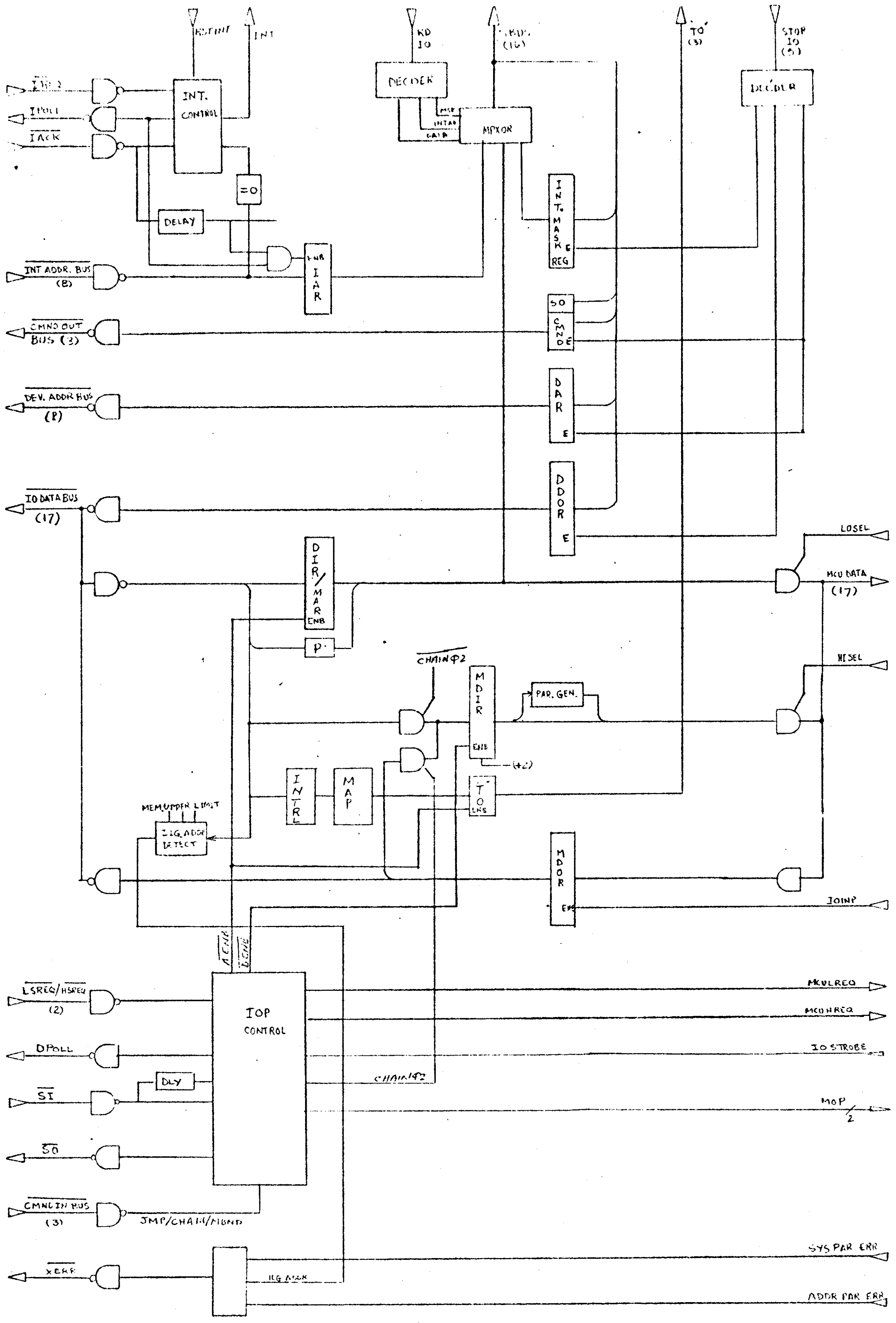
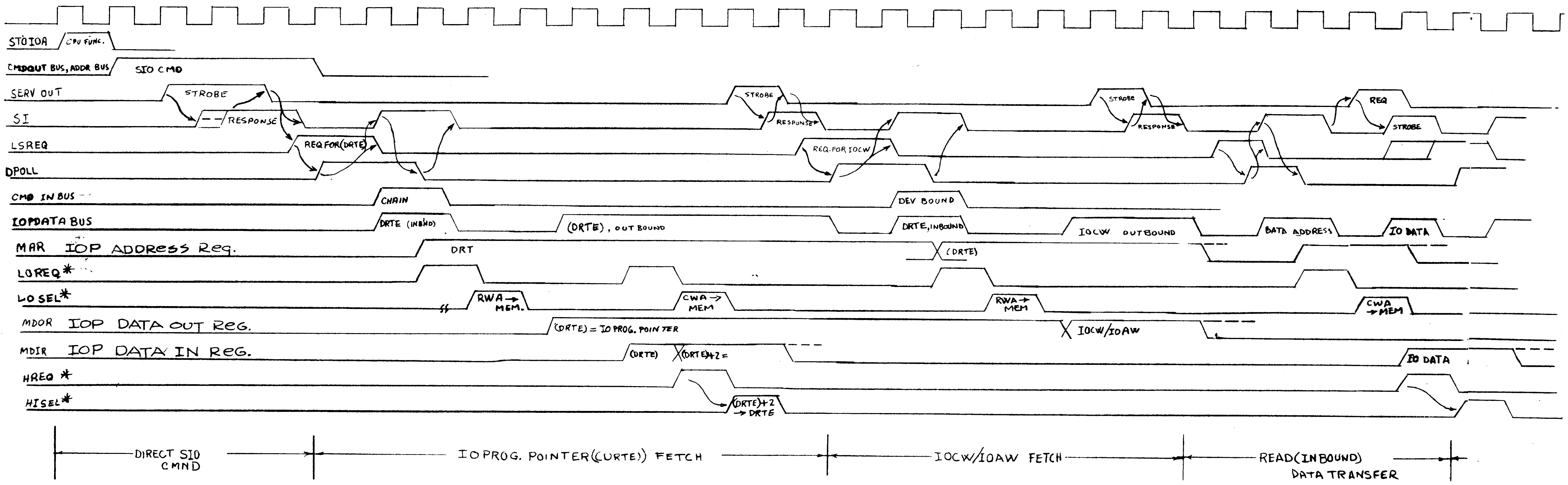


Figure 8. IOP Functional Block Diagram



* LOREQ - IOP REQUEST TO TRANSFER ADDRESS TO MEMORY
LOSEL - MCU BUS CYCLE TO TRANSFER THE ADDRESS TO MEMORY
HREQ - IOP REQUEST TO TRANSFER DATA TO MEMORY
HISEL - MCU BUS CYCLE TO TRANSFER THE DATA TO MEMORY
DRTE ≡ DATA REFERENCE TABLE ENTRY

Figure 9. IOP Bus Timing

SECTION 4

I/O TERMINATING TECHNIQUES

Some preliminary tests have been completed on the IOP bus and the SIO bus. To minimize crosstalk and settling time, the following recommendations for bus terminations are made.

4.1 PROTECTED CIRCUITS

The accepted method of using 50-conductor flat cable, 3M #3365, is to have each circuit protected with an adjacent ground return lead. The so-called protected circuits are found on edge sensitive signals such as $\overline{\text{CHAN SO}}$, $\overline{\text{CLOCK}}$, $\overline{\text{REQ}}$, etc. They are all unidirectional, either originating at IOP or SIO and driving the bus, or wired-or at each device controller and terminating at the IOP or SIO board.

When properly terminated with 100 ohms the pulse fidelity is excellent. When mismatched with a 200-ohm termination, the negative going edge will ring, creating a +0.3v noise oscillation peak after 50 ns. A Schottky diode to ground (as will be found in the 106A/B receivers when they are installed) effectively clamps the oscillation. The positive edge ringing is not troublesome.

The recommended termination is thus a 200-ohm resistor to +3.6v. See figure 10.

For signals originating at the IOP or SIO board, the terminating network will be built on the Terminator Board at the end of the backplane. For the wired-or signals, the terminator will be on the IOP or SIO board itself.

4.2 UNPROTECTED CIRCUITS

Many of the bus signals are unprotected, notably the 16 data leads, DEV ADDR, INT ADDR, etc. These do not have ground returns between the signal leads. Also, some are bi-directional and at times all transmitters may be disabled.

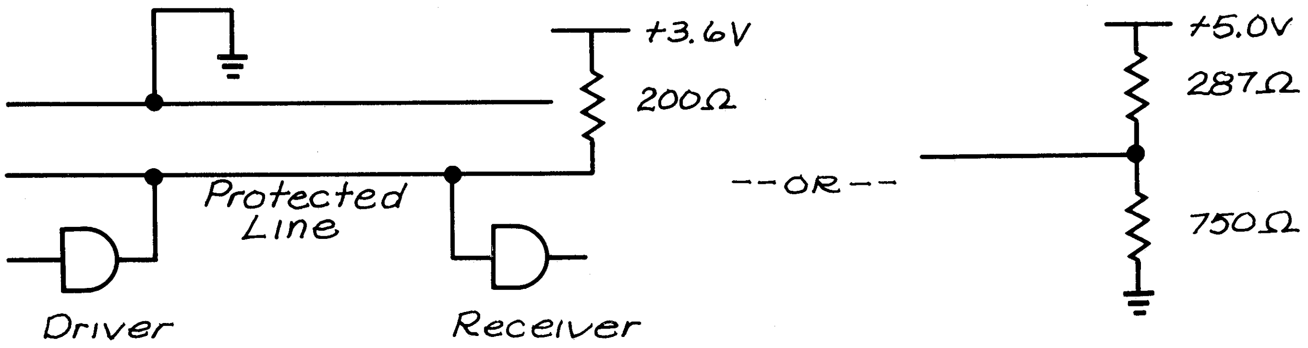


Figure 10. Terminations for Protected Lines

Due to inductive coupling between the leads, it was found that almost any current in the circuit caused crosstalk. The worst case being where all signals were changing in the same direction.

The standard 200-ohm terminator is not suitable for unprotected circuits. Crosstalk lasting over 500 ns was observed.

HP 104 drivers or TTL gates are not suitable as bus receivers. The TTL drive current per input caused 1/2v of crosstalk in the I/O simulator with only a single device controller plugged in. The total crosstalk causing drive current in this example being 8 ma.

The minimum possible current in the circuit is that required by the 106 receivers when the line is in the high state and the 104 drivers in the disabled state. The 106 receiver drive is 100 μa and the driver leakage is 40 μa.

Tests have been run with 3 ma of drive current representing 30 HP 106 receivers on the line. The ribbon cable length was 6 feet. Crosstalk was 0.1 to 0.2 volt, close to the measurement limit. The signals were entirely satisfactory, settling in less than 30 ns. The recommended terminator is shown in figure 11.

The negative going overshoot is caught by the Schottky diode input of the 106 receivers, the positive overshoot is caught by the diode on the terminator board.

Note that the state of the line is undefined when all transmitters are disabled.

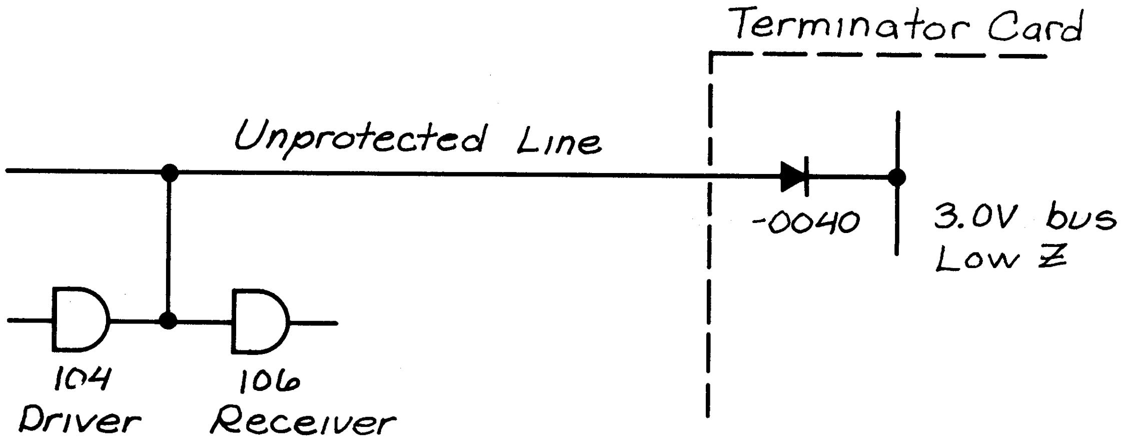


Figure 11. Terminations for Unprotected Lines

4.3 UNPROTECTED EDGE-SENSITIVE CIRCUITS

A third class of signals is found on both the IOP bus and the SIO bus. These are handshake type edge-sensitive signals that are not protected by adjacent grounds. Examples are $\overline{\text{DEV END}}$, $\overline{\text{ENABLE}}$, $\overline{\text{RD NEXT WD}}$, etc.

These signals are wired-or with pull-up resistors to Vcc. It is not possible to use 3-state gates and eliminate the pull-ups since the state of the line must be defined at all times. We are thus forced to live with the crosstalk.

Two factors save us:

(a) In general, only one line is asserted at a time.

(b) The high state (the 0 state) can be defined to be +3.6v, this affording us an additional 1.2v of noise margin.

The recommended receiver is thus the same as that shown in part 1, above (200 ohms to 3.6v). Tests over the 6-foot ribbon cable have shown a noise pulse of 0.7v (3.6v to 2.9v) when one of the lines is asserted.

Note

The drivers must be well bypassed. A 0.1 MFD capacitor by each 104 DIO is recommended.

SECTION 5

INTERFACE CONSIDERATIONS FOR DEVICE CONTROLLERS

Most device controllers communicate with the ALPHA via the bus logic and IOP bus. Each device controller is an individual case and to try to describe each device controller would be prohibitive. Some device controllers will have command registers, some will not. Some device controllers will have a status register; some will not. Some will have data registers; others will rely on the data register within the device themselves.

There are two sets of strobes emanating from the bus logic interface. These are the DIRECT strobes (generated by direct I/O instructions) and the PROGRAMMED strobes (generated by I/O orders within an I/O program). There are four strobes in each group:

Data In

Data Out

Status In

Command Out

These devices which utilize the direct I/O commands only will use only the direct strobes. Those devices which may execute I/O programs (SIO type) may use both sets of strobes.

The SIO type device can 'or' the strobes of the same type (i.e., the direct Status In and the Programmed Status In may be ORed together to produce a single status strobe). Or the strobes may be kept separate. For example, a device controller may have two 16-bit status registers, one which is read by the I/O program and the other which is read by the CPU by a TIO command.

Some devices perform only one simple operation. When they receive a data word they perform some operation on it (e.g., a D to A converter). For such devices, a command register would be unnecessary. However, most devices will require a command from the ALPHA to tell them what to do. Figure 12 depicts the command register and the associated gating.

The command strobe (which may be the logical OR of the direct and programmed strobes) clocks the 16-bit command register consisting of four quad latches or equivalent. The 16 data-out lines contain the command during the time the command strobe is asserted (remember data-out lines are plus-true). The command strobe is shown going to the device controller in figure 12. The device controller may set a flip-flop, or perform some other operation when the command strobe

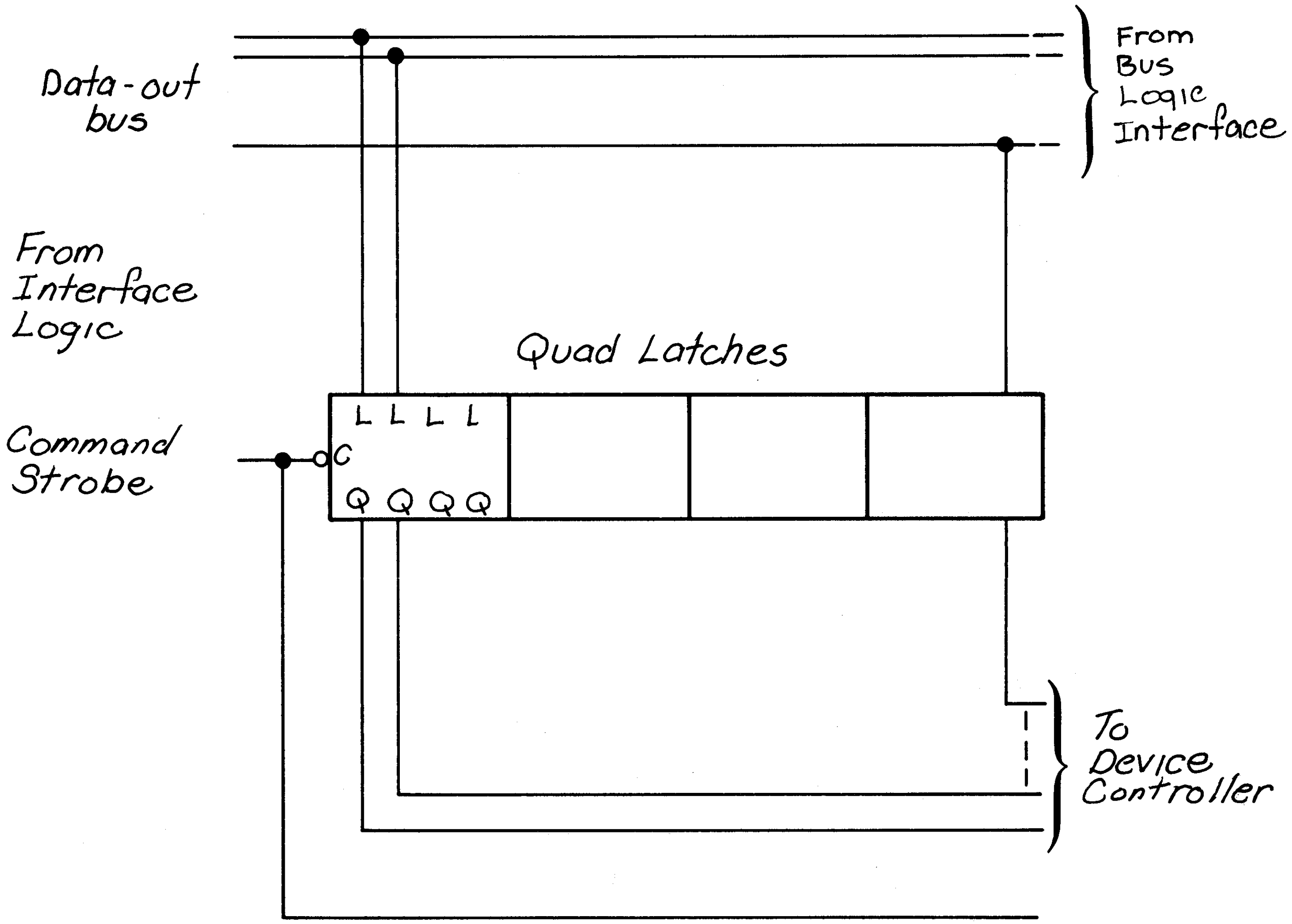


Figure 12. Command Register Gating

is generated. That is the command strobe not only strobes the command into the command register, but is also may signal the device controller that a command has been generated.

Commands may be used to establish initial conditions in a device controller, to set the device controller into a particular mode, to tell it whether to read or write, etc.

The status register is used to convey to the program the status of the device and/or device controller. Figure 13 depicts the status register and its associated gating. The Status Strobe gates the contents of the status register onto the data-in lines. Since some of the status lines from a device are static in nature, these need not be stored in a flip-flop register. They may simply be gated onto the data-in lines. An example of such a status line is shown in figure 13.

The status register may contain any information which may be informative to the program. Such information may include reason for error interrupt, current mode of the device, file protected, card jammed, on-line/off-line, etc.

Data registers may reside either in the device controller or in the device itself, depending on the specific device involved.

The gating is similar to the gating shown in figures 12 and 13 for commands and status. Again the Data-in and Data-out strobes may be used by the device controller as "proceed" signals.

Of course it is up to the device controller to assure that data is valid before it requests service from the ALPHA to input the data word. Or the device controller must take the data from the ALPHA before requesting another word.

5.1 INTERRUPT INTERFACING

The interface to the Interrupt Logic portion of the bus logic interface is extremely simple. Figure 14 shows this interface. The four lines from the SIO Interface are SET INT, RESET INT, DEV INT and INT ACTIVE. The device controller designer MUST connect these three lines as shown in figure 14. These lines are connected to the direct set, direct reset and $\overline{Q}$ terminals respectively of either a D/flip-flop

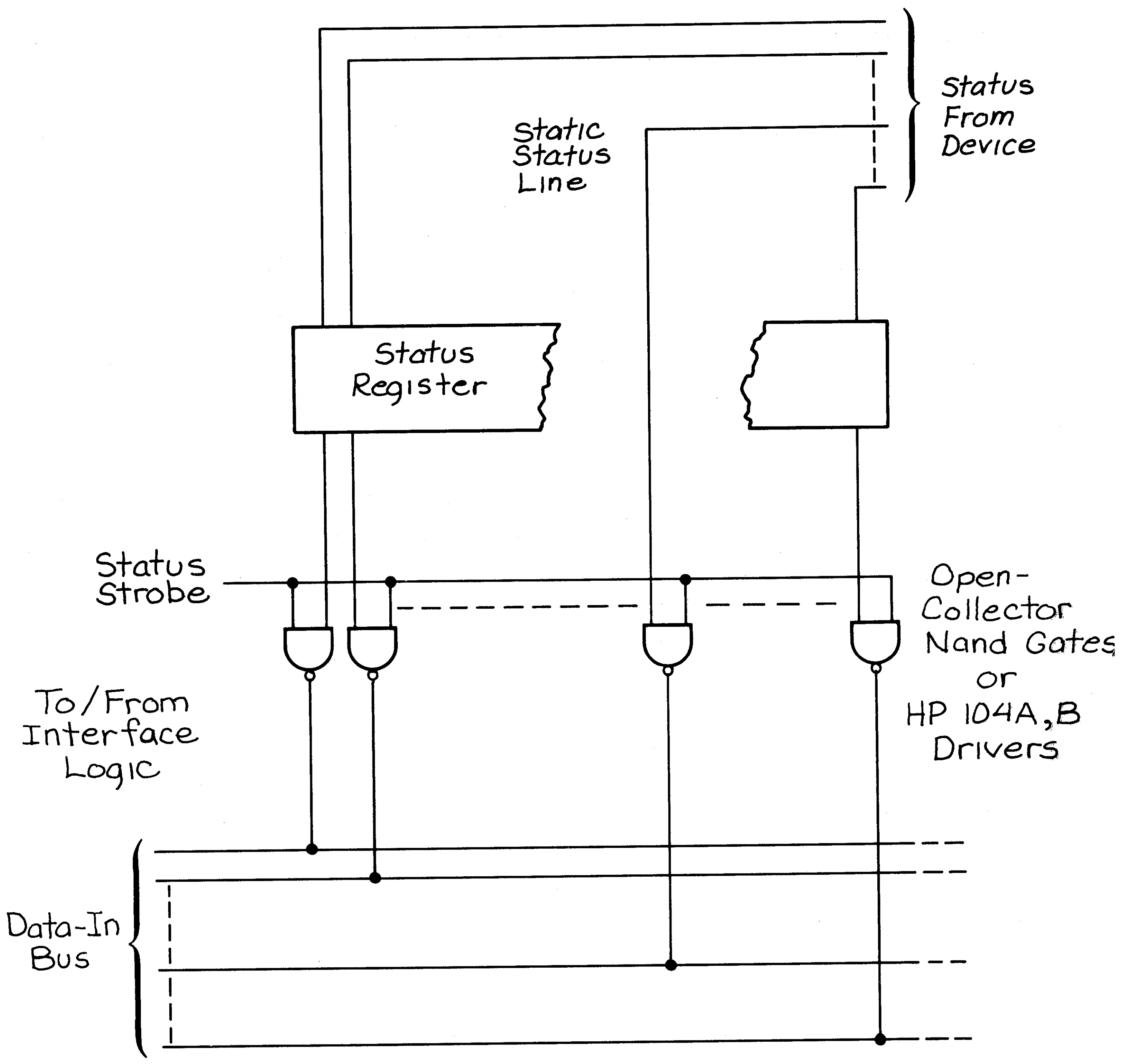


Figure 13. Status Register Gating

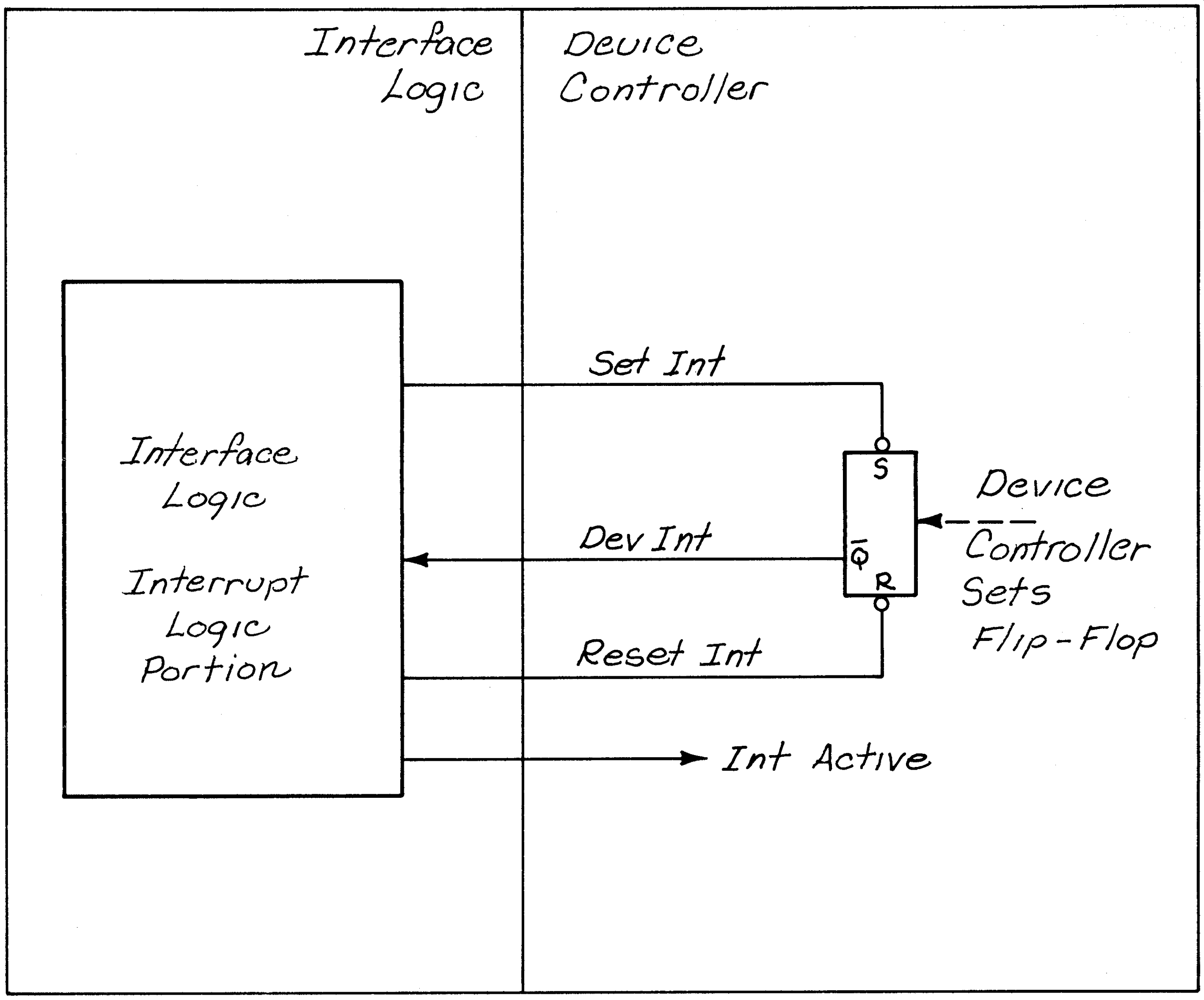


Figure 14. Interrupt Interface

or a J-K/flip-flop. The device controller may set the flip-flop only by the edge sensitive clock input of the flip-flop (and of course, the D or J-K terminals). No attempt should be made to direct set the flip-flop because the RESET INT signal may occur while the active LOW SET is asserted.

The interrupt circuitry is independent of the data transferring circuitry of the Interface. Hence, setting the interrupt flip-flop does not terminate or in any other way affect the execution of the I/O program.

The SET INTerrupt line allows for a programmed interrupt or a CPU generated interrupt. The RESET INTerrupt line resets the device's interrupt flip-flop at the time the interrupt route is exiting.

INT ACTIVE is asserted for the duration of the interrupt processing, i.e., from IPOLL to RESET INT. The device controller Interrupt FF is reset when the IPOLL is received by the requesting device controller.

5.2 I/O BUS LOGIC

To help the device controller designer interface with the IOP Bus and the multiplexed SIO card or Selector Channel, three versions of bus logic are provided:

(a) Direct CPU Commands only

(b) Multiplexed SIO

(c) Multiplexed SIO or Selector Channel

Both (b) and (c) provide the direct CPU capability.

The functions of the bus logic are listed below:

1. Decode direct CPU commands to provide the direct strobes, responding to $\overline{SO}$ by returning $\overline{SI}$.

2. Initiate the CPU external interrupt by asserting $\overline{IREQ}$ to the IOP, and provide the handshaking required to transfer the device number to the IOP in response to IPOLL.

3. Generate a $\overline{REQ}$ to start an I/O program to the multiplexed SIO card or Selector Channel in response to the direct CPU command SIO.

4. Buffer the Programmed Strobes generated by the multiplexed SIO or Selector Channel in response to an executing I/O program.

In most cases the interfacing signals are ground true when asserted. The data buses (DATA IN bus and DATA OUT bus) can be defined by the device controller designer remembering that the data bus in the IOP bus is ground true. The inverting and non-inverting drivers and receivers can be used to provide either sense for the interface to IOP bus, thus providing maximum versatility for the designer. The signal XFER ERROR is positive true from the bus logic interface. The CLK XFER ERROR signal is also positive true from the bus logic interface.

In the discussion "asserted", "true" and "active" will all mean a ground true signal unless otherwise noted.

The signals are all TTL type signals.

5.2.1 Master Reset

This signal is generated from the IOP bus signal IORST. It is ground true and can be used by the device controller to reset any of its various states. The bus logic resets with the signal Master Reset to an idle condition.

5.2.2 Direct Strobes

These are READ, WRITE, STATUS, CMD. These lines are asserted when a RIO, WIO, TIO or CIO instruction is executed by the ALPHA. The direct strobe will remain true until the device controller acknowledges by asserting the ACK line.
The DATA IN and the STATUS strobes should gate the contents of the data in register and the status register respectively onto the DATA IN bus. The DATA OUT and CMD strobes should strobe the DATA OUT bus into the data out register and the command register respectively. Either edge or active low may be used.

5.2.3 SERV RQ

This line should be asserted by the device controller when a) it has completed the execution of the last programmed Command issued, b) when it has data ready to be transferred to memory (the data should already be in the device controller's data in register), c) when it wants a word from memory, or d) when the IN XFER or OUT XFER has gone false (word count has gone to zero) and the device is ready for the next order, e) following the first word of programmed control information. The SERV RQ line is used only for SIO type devices. The I/O program will not continue following execution of a Control, Read or Write order until the SERV RQ is asserted.

The SERV RQ must be held true by the device controller until the SIO Interface asserts the SR ACK line. SERV RQ must not be asserted again until SR ACK goes false. See Selector Channel discussion for the specifics when connected to the channel.

5.2.4 SR ACK

This line is the multiplexed SIO response to SERV RQ. SERV RQ must be held true until SR ACK is asserted, and must not become true again until SR ACK goes false. The Selector Channel does not generate SR ACK.

5.2.5 DEV END

This line is asserted by the device controller only when an I/O programmed Read or Write is being executed by the controller. This line terminates the block transfer. The DEV END should be used by devices which read variable length records (paper tape, magnetic tape, for example) or by devices which terminate a write operation before the word count has gone to zero. This line should be asserted before the device controller would normally assert the SERV RQ; SERV RQ need not be asserted by the device controller but there is no harm in doing so. DEV END must be held true by the device controller until the SIO Interface asserts SR ACK.

5.2.6 OUT XFER and IN XFER

One of these lines will be asserted by the bus logic interface when a block transfer is to take place (SIO type device only). It will remain true until the word count has gone to zero in the WC register AND the last word has been transferred. This line will remain true during data-chaining. The data-chaining will not be apparent to the device controller. Needless to say, OUT XFER will be true for a Write order and IN XFER will be true for a Read order.

5.2.7 SET JMP CONDX and JUMP

During an I/O program a conditional jump may be encountered. When this occurs a pulse will be generated by the SIO Interface on the SET JMP CONDX line. The device controller should use this line to clock a flip-flop which should set if a jump condition is met and reset if not. The $\overline{Q}$ of this flip-flop should be sent to the bus logic on the JUMP MET line.

The question is: what determines whether a jump condition is met or not? This is up to the device controller designer. For example, a special command might contain a mask which will determine which condition(s) will cause a jump. For instance the mask could enable or disable a jump on parity error, jump on hopper empty, jump on end of tape, jump on file mark, jump on track not found, etc.

5.2.8 ACK

There are actually four ACK lines, but they are logically ORed in the bus logic. The ACK must be the immediate response to any of the strobes, direct or programmed. Where the device controller has all of the data, status and command registers locally the strobes may merely be turned around and sent back as ACK signals. Since the entire I/O system will be hung-up waiting for the ACK, it is imperative that the ACK not be delayed any more than necessary.

5.2.9 DATA OUT Bus

This 16-line bus provides the command and data out path for both direct and programmed data transfers. It contains valid data only when appropriate strobes are asserted. Since this bus is shared by all device controllers its state cannot be predicted or guaranteed except during strobe time.

The DATA OUT bus may be used to drive D flip-flops or latches (D or L terminals respectively) or may be gated to set-reset flip-flops.

5.2.10 Programmed Strobes

These are DATA IN, DATA OUT, STATUS, CMD, PCMDI. These lines are asserted respectively when the device is executing an I/O programmed Read order, Write order, Sense order or Control order. The strobe will remain true until the device controller acknowledges by asserting the ACK line. The DATA IN and STATUS strobes should gate the contents of the data in register and status register respectively onto the DATA IN bus. The DATA OUT and CMD strobes should strobe the DATA OUT bus into the data out register and command registers respectively. Data is not valid at the leading edges of Write and CMD when strobes are generated in the Selector Channel.

The direct strobes and the programmed strobes may be ORed or remain separate, depending on the particular device controller.

5.2.11 DATA IN Bus

This 16-line bus provides the status and data in path for both direct and programmed data transfers. Data should be gated onto this bus <u>only</u> when the appropriate strobes are asserted.

5.2.12 Clear Interface

The CL INTFC line should be pulsed by the device controller when it aborts an I/O program. This would be the case when, for example, an error occurring during execution of an I/O program is catastrophic enough to warrant abortion of the program and a CPU interrupt to be generated. This signal should not be confused with DEV END which merely terminates a Read order or Write order, but continues the I/O program. This line will be asserted in response to a XFER ERROR signal.

5.2.13 SIO OK

This signal, when asserted, means the SIO interface is in the idle state. It must be returned to the SIO Interface as status bit 0. The device controller may logically AND this signal with its own "device ready" type of signal, and then return it as status bit 0. In this way the ALPHA is guaranteed not to execute an SIO instruction unless both the SIO Interface <u>and</u> the device are ready. Similarly, bit 1 of the status word must be used by all device controllers to indicate that either a WIO or RIO shall be permitted. The SIO OK signal need only be used for SIO type devices.

5.2.14 Power On Pulses

These are discussed previously in the IOP bus definition.

5.2.15 Read Next Word

The RD NEXT WD line is asserted by the SIO Interface only during a programmed Read order. It tells the device controller to fetch another word from the device. This line should be useful for such devices as card readers and paper tape readers. This signal will occur simultaneously with the programmed DATA IN Strobe.

Note

During data transfers to and from memory an error condition may arise due to:

Illegal Address

Memory Address Parity

Data Parity

If this happens during an SIO program it may be necessary to return the error condition to the CPU as a part of the status word. The signal XFER ERROR will be asserted (pulled low) if an error has been detected during Data time. It may be necessary to store the information in a flip-flop, return the information as part of the STATUS WORD and then clear the flip-flop. See figure 15.

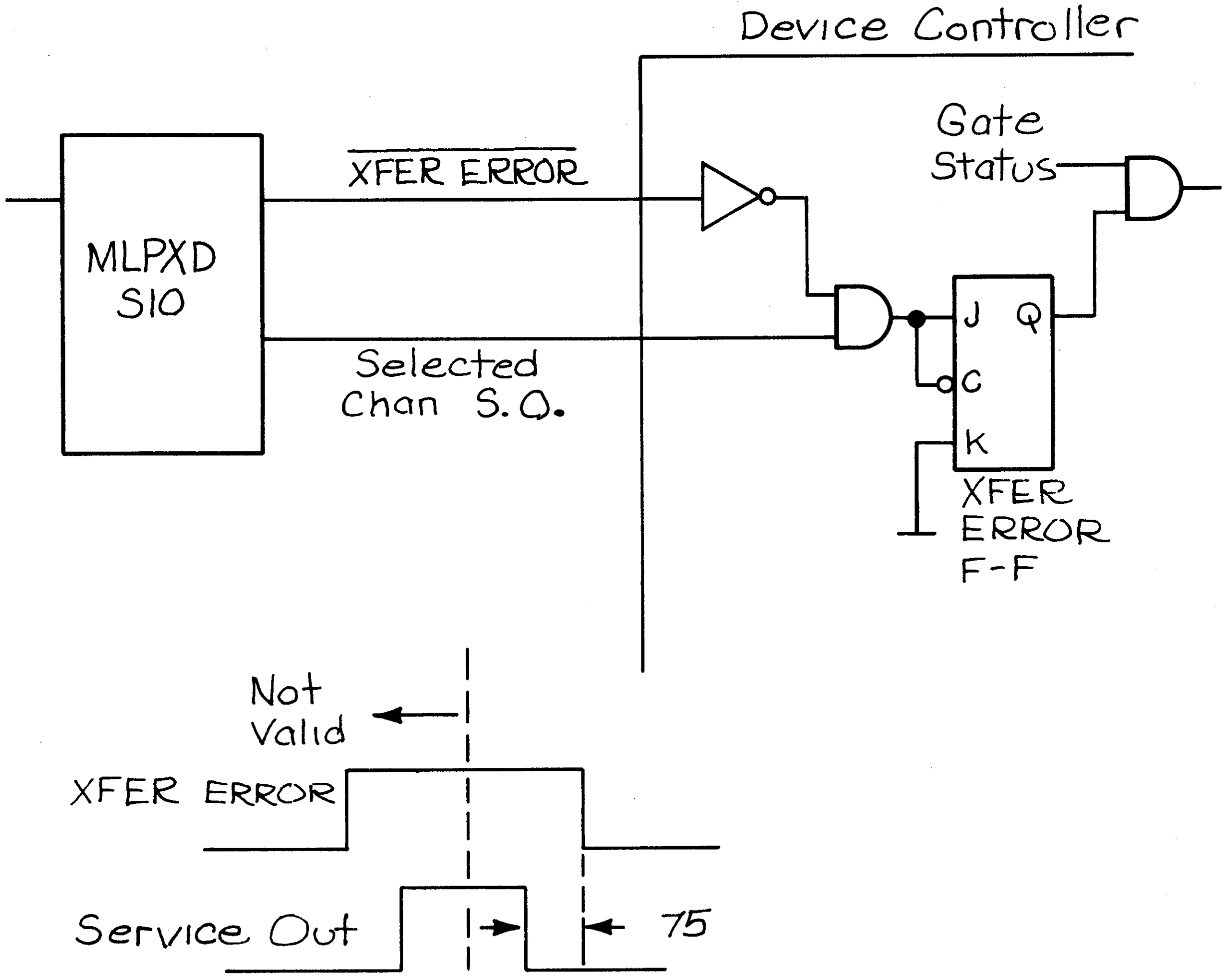


Figure 15. Transfer Error Logic

The trailing edge of Service OUT must be used to CLOCK the XFER ERROR line. A device controller must respond to a true error condition by:

1) Asserting Clear Interface and terminating the I/O program

2) Generate Interrupt

3) Set status bit to indicate that the condition has occurred.

The bus logic provides a clock signal to be used for clocking the XFER ERROR line. This line is called CLK XFER ERROR and it is positive true.

See the Multiplexed SIO and Selector Channel discussions for more specific information and timing considerations for the signals and strobes discussed here briefly.

SECTION 6

ALPHA INTERRUPT SYSTEM

This section provides the device controller designer with some insight as to the response of the Interrupt Request generated by the bus logic made by the Alpha interrupt processing system firmware.

6.1 INTRODUCTION

The Alpha interrupt processing system is a combination of hardware and firmware used to control the transition to and from software modules needed to process I/O demands and various internal CPU events. The hardware is used to sense the need for an interrupt and inform the firmware of what action to take, while the firmware actually does all the processing necessary to transfer to and from an interrupt segment.

Interrupts may be separated into two distinct types, internal and external, and each is handled in a different manner.

6.2 INTERRUPT STACK

All the software interrupt modules whose segment number is less than 8, use the Interrupt Control Stack (ICS) for processing interrupts. Thus, the firmware not only changes the program segment, but also the stack area. The ICS and associated pointers are shown in figure 16. QI and ZI are pointers which locate the bottom and top limits of the stack. The bottom of the stack is capped off with a dispatcher stack marker put there at cold load time and retained permanently. It has special significance which will be shown later.

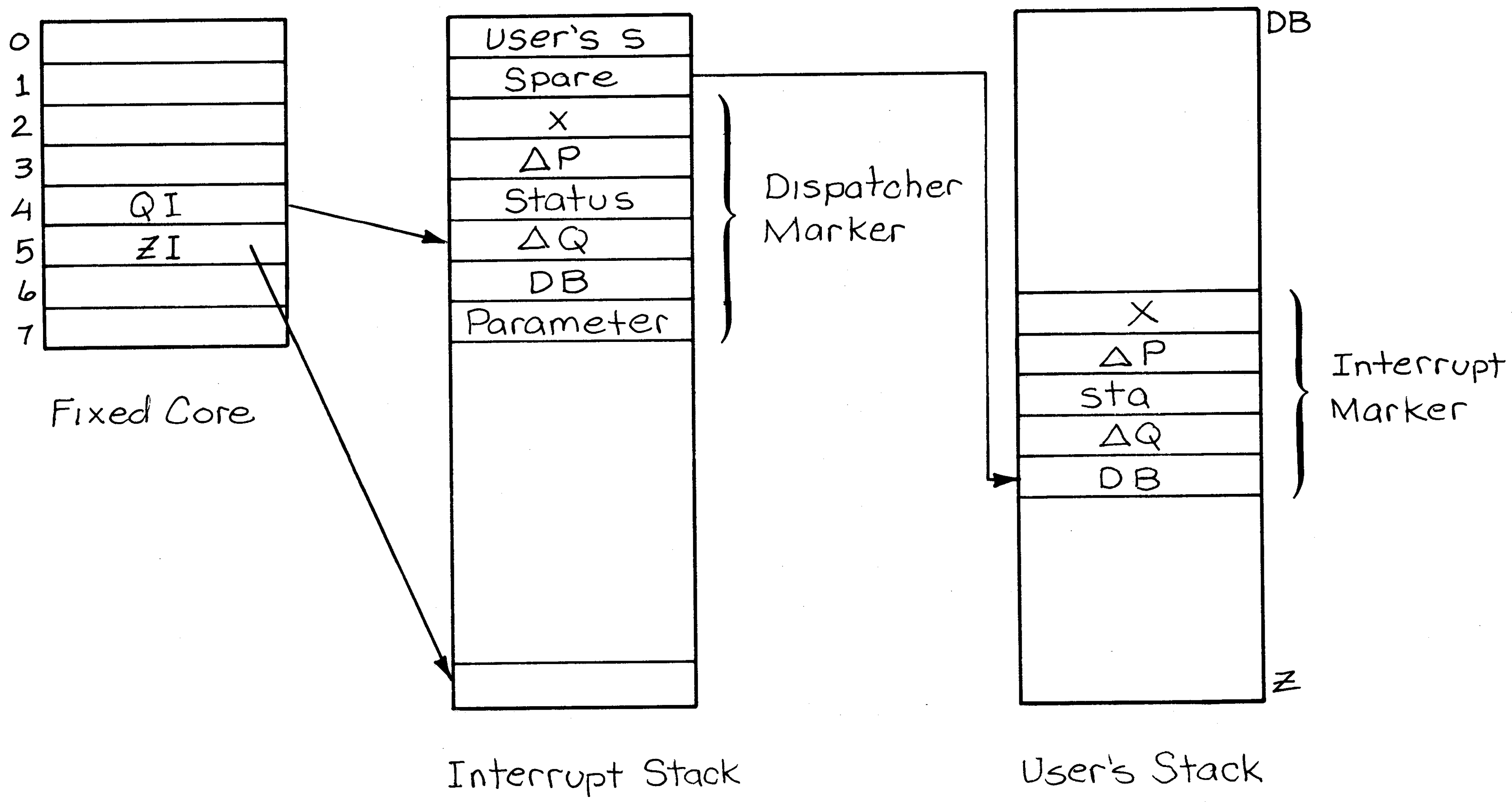


Figure 16. Core Map After Interrupt

6.3 INTERRUPT PROCESSING

Interrupt processing may be separated into two types, external and internal. External types will be discussed first. Figure 17 shows the overall flow diagram of the interrupt processor.

During a currently executing instruction the CPU hardware detects the need for an external interrupt via the IOP. At the end of the instruction, microcode control is transferred by hardware to microcode location 2 in ROM. This starts the interrupt processor. From here, everything is handled by microcode. Looking into the run state interrupt register (CPX1), microcode determines that this is an external interrupt and stores a 0 for segment number in one scratch register, and the device address from the IOP interrupt address register into another scratch register. Then control is passed to a microcode program used by all interrupts who need to transfer to the interrupt control stack. (Segment Number < 8).

A standard four word stack marker is pushed onto the stack, followed by the current value of the DB register. (DB will change). The CPU knows if it is on the interrupt stack through the use of a hardware flag (ISF). If ISF = 0, then control is transferred to the interrupt stack, while ISF =1 bypasses the transfer to maintain the stack integrity. This flag is reset by the SETR instruction when Z is set to a value other than ZI (as when control is passed from the ICS back to the users stack). To transfer to the interrupt stack from a user stack, Q is set to QI, Z is set to ZI, the current value of the user's S is saved at QI - 5, and S is set to QI + 1 (to preserve the dispatcher marker). The ISF is set to 1, and the dispatcher flag (DF) is set to 0. Then, the interrupt parameter, which is the device address, is pushed onto the stack. Now that the stack transfer has been completed, the remaining task is to set up the program environment and, because this is an external interrupt, the DB register. The program limits are set to the memory extremes, $PB \leftarrow 0$, $PL \leftarrow 2^{16} - 1$. Status is set to 140000, meaning privileged mode and interrupts enabled. The P register is set to PBI and DB is set to DBI from the device reference table in lower memory. This completes the program transfer.

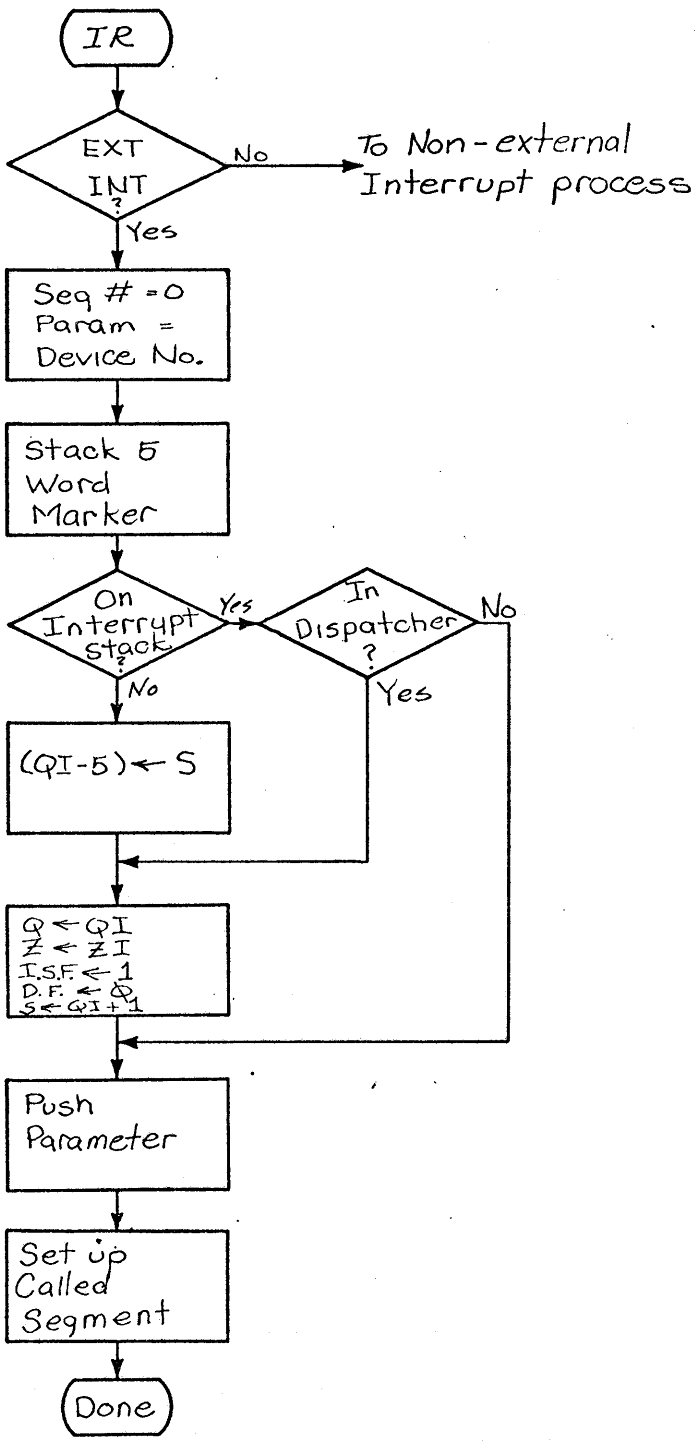


Figure 17. Partial Flowchart, Interrupt Processor

After completion of the external interrupt routine, it returns to the operating system environment via the EXIT instruction. In the case of an external interrupt, the EXIT microcode will reset the active interrupt state of the device whose device number is located at Q + 2 (this device number is the parameter put at Q + 2 during the interrupt transfer), which means that the I/O interrupt program must retain the contents of location Q + 2 during execution. If the device cannot be reset, the computer will halt. After resetting the device, EXIT will look to see if any lower priority devices are waiting to interrupt. If there are, then EXIT will go to the interrupt processor and set up the desired interrupt routine. This saves doing a full EXIT followed by another full interrupt process. If no other interrupts are pending, EXIT will complete as a normal exit.

Internal interrupts are essentially handled as a PCAL instruction except that segments one through seven transfer to the interrupt stack and push DB, and that most interrupts pass a parameter on the top of the stack. An EXIT from an interrupt is like any normal exit.

6.4 THE DISPATCHER

The last segment to run on the ICS exit to a software segment is known as a dispatcher. This segment determines what has happened since it was last run, and what environmental changes need to be performed at this point, such as allocation of system resources, swapping of programs on and off the disk, and start up of programs.

The exit microcode is made aware of an exit to the dispatcher by the fact that the stack marker Δ Q is zero. It sets a hardware dispatcher flag to keep track of the fact that the dispatcher is executing. This flag is reset in the same manner as the interrupt stack flag. If the dispatcher is interrupted by an external interrupt, the dispatcher program is aborted by handling the interrupt as if it happened on the users stack, except that the current value of S is not saved.

The I/O interrupt program can make the dispatcher aware that it has serviced an interrupt by setting the IRF bit in its DRT table with the SIRF instruction.

SECTION 7

MULTIPLEXED SIO LOGIC

Figure 18 is a block diagram of the SIO card. It can be viewed as three separate machines working in parallel. The first machine, or portion of the logic, executes the SIO programs. This requires the unloading from and storing into the RAMs, incrementing the order and address registers, determining the next state, and generating control signals to the device controllers.

The second portion of logic is the parity control logic. It checks parity on each transfer to the SIO card, transmits odd parity with each address sent to the IOP, checks each output from the state RAM, and will generate an XFER ERROR signal to the device controller whenever it detects an error. This logic is intended to prevent SIO programs from executing outside their actual program area.

The third portion of logic included in the SIO design is the diagnostic controller. This logic is not indicated in figure 18, but is included as part of the control logic. It will allow all but the parity RAM to be loaded individually, and each of the RAMs and registers to be read separately. The address and order registers can be incremented with or without first being loaded from their respective RAMs. In addition, the next state function of the SIO control logic can be tested by first loading the order and state RAMs and then performing successive reads from the state RAM. These diagnostic features have been included to allow nearly complete verification of the operation of the Multiplexed SIO card by the CPU when in a bottom-up mode.

The following sections will cover each of the three sections discussed above in somewhat more detail.

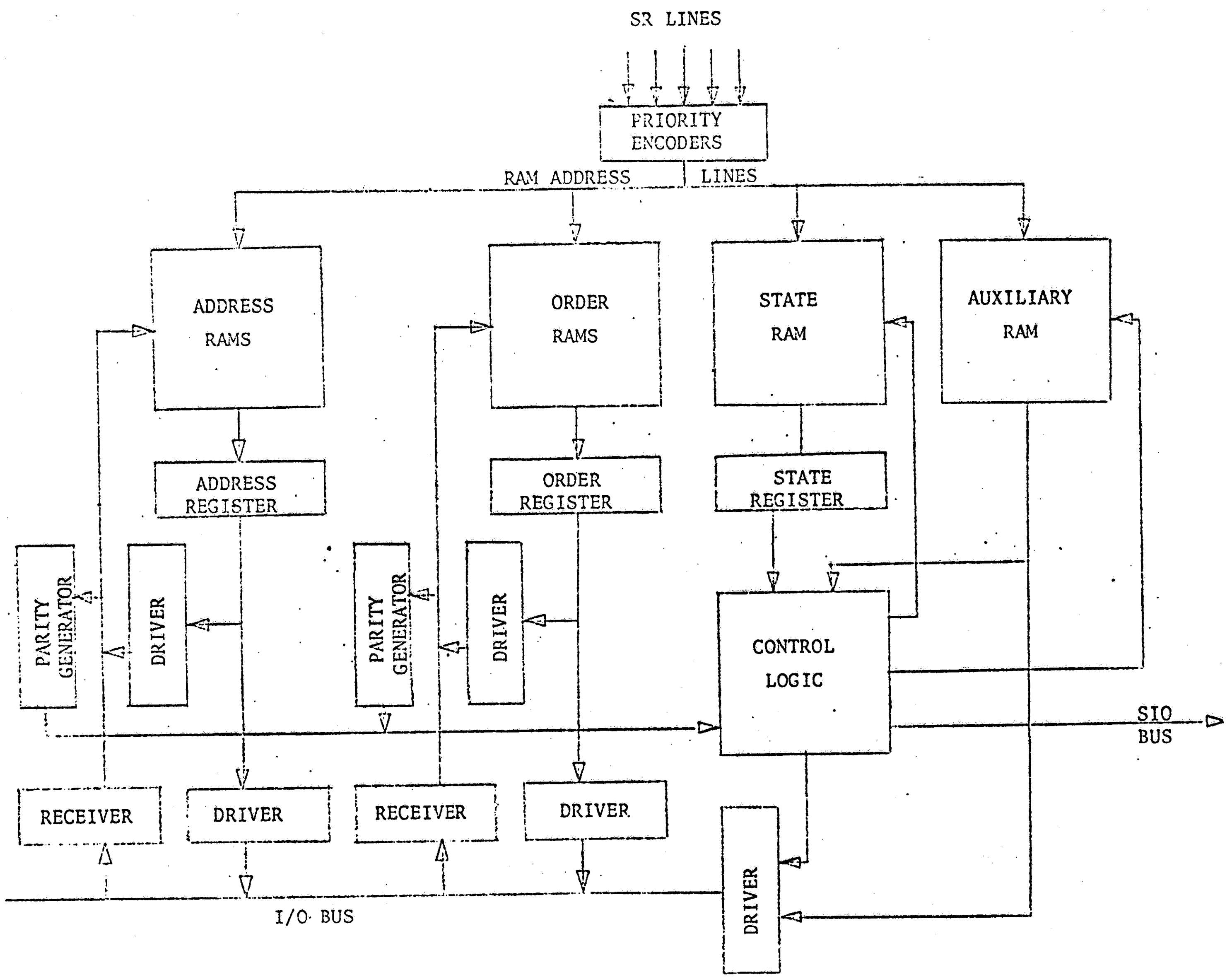


Figure 18. Multiplexed SIO Card

7.1 SIO PROGRAM CONTROL LOGIC

The SIO Control Logic has the capability of executing 8 I/O orders:

1. Read
2. Write
3. Control
4. Sense
5. Return Residue
6. Interrupt
7. Jump
8. End, with optional interrupt

The word format for each of these orders is given in figure 19. In every case, the first word of each order is stored in the order RAM, while the second word, if needed, is stored in the address RAM. Only the JUMP address or the starting address of the data block of a READ or WRITE is ever actually fetched and stored into the address RAM. The second word of a CONTROL order is always fetched, but is gated to the appropriate device controller to be used as control information and is not loaded into the address RAM. The second word of the INTERRUPT order is always fetched, but is disregarded.

The second word of the SENSE, RETURN RESIDUE, and END orders is used for storage of information from the I/O system. In the case of the SENSE and END orders this second word will be loaded with status information sent from the device controller, while the RETURN RESIDUE order will cause the contents of the SIO card's order RAM to be stored.

As mentioned earlier the SIO card time division multiplexes the execution of up to 16 I/O programs. During each time slice an address is sent to memory, which is then followed by data to or from that address.

All transfers on the I/O bus are asynchronous and are controlled by a handshake. Initiation of each address-data transfer is done by the device controller by generating

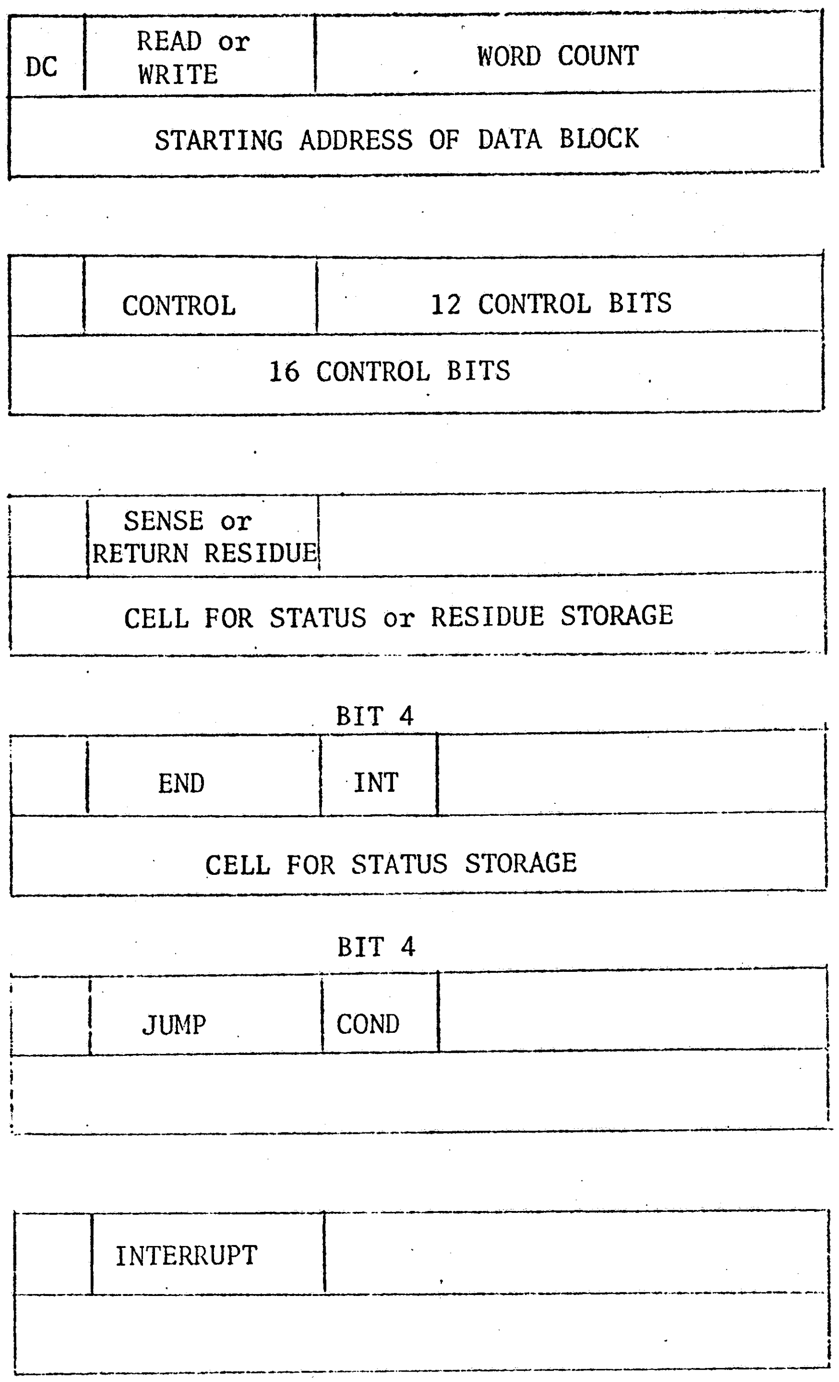


Figure 19. I/O Command Word Formats

a service request to the SIO logic. If the SIO logic receives any service requests it will, in turn, request service from the IOP. The IOP then responds with a data poll, which is daisy-chained from one SIO card to the next. The highest priority SIO requesting service will stop the poll and proceed through an address transfer sequence.

An address transfer sequence consists of 3 major segments:

1. Securing a valid RAM address
2. Unloading RAMs and generating a select code to the device controllers
3. Returning an address and command to the IOP with the response to the data poll (Service In)

A detailed description of the address transfer sequence is given in Section 7.4 and the associated timing diagrams can be found in Section 7.4.1. It will suffice to say here that the actions to be taken by the IOP during the data transfer sequence are determined by a command sent to the IOP from the SIO logic during the entire address-data transfer.

If the DRT entry is to be sent by the IOP to the SIO logic, the IOP will increment the entry by 2 after it is transferred and then restore it in memory.

Should the command to the IOP be a JUMP, the IOP will increment the address sent to it during the data transfer sequence by 2 and then store it into the DRT.

In both of the above cases the DRT entry to be affected is determined by transferring the device number during the address transfer sequence. By placing the device number on bits 6-13 of the IOP DATA bus the device number is mapped into the correct DRT entry of four times the device number.

In the case of data into memory from the I/O devices, the IOP will merely transfer the data it receives during the data transfer sequence to the memory address it receives during the address transfer sequence.

Finally, data out, from the address received by the IOP during the address transfer sequence, is sent to the I/O device during the data transfer sequence.

A detailed description of the data transfer sequence can be found in Section 7.4.2, with corresponding timing diagrams in Figures 22 through 30. The portion of the data transfer sequence that is of concern here is the control signals sent by the SIO logic to the selected controller during the transfer.

If the SIO logic is executing either a SENSE or END a Programmed Status Strobe will cause the selected device to gate its status to the IOP DATA bus, and hence to memory through the IOP.

Execution of a WRITE order will cause a Programmed Write Strobe to be issued to the selected device controller. The controller uses the signal to strobe data into a holding register from the IOP DATA bus. Transfer of the data to the device is then under control of the device controller.

The READ order execution will proceed similar to the WRITE, but the Programmed Read Strobe signal will gate data from the controller's holding register onto the IOP DATA bus. At the same time the Programmed Read Strobe is issued another signal, READ NEXT WORD, will be issued. This signal can be used by device controllers to initiate the transfer of the next data word from the device to the controller. As is apparent, the READ NEXT WORD signal will precede the Programmed Read Strobe by one pulse. That is, one RD NEXT WD pulse will occur before the first Programmed Read Strobe, and the last Programmed Read Strobe will not have an accompanying RD NEXT WD signal.

The execution of the CONTROL order will generate 2 signals – Program Command 1 and Programmed Control Strobe. At the same time the SIO logic loads the CONTROL order into the Order RAM, it will generate the Program Command 1 signal to the selected device controller. This will inform the controller that the 12 least significant bits on the IOP DATA bus – bits 4 through 15 – can be used as control information.

Programmed Control Strobe will be issued by the SIO card whenever it fetches the second word of the CONTROL order, and, as explained earlier, the information on the IOP DATA bus will be used by the selected controller as control information.

By creating the 2 control signals, device controllers can optionally use up to 28 bits of the CONTROL order for control information. This will reduce the number of CONTROL orders needed for complex device controllers such as the ISS moving head disc controller. This, and other capabilities, will reduce the dedicated memory required for I/O driver programs.

Execution of the INTERRUPT order will generate a SET INT signal to the device controller, forcing its interrupt flip-flop to set. This can also be done during execution of the END order by setting bit 4 to a one in the order word.

Execution of the JUMP order will cause one signal to be sent to the device controller and one response to be returned. At the time the SIO logic receives the jump address from the IOP, it will send the SET JUMP signal to the selected device controller. This signal will be used to clock the controller's Jump flip-flop. The inputs to this flip-flop are determined solely by the controller designer, but their function is to permit conditional jumps in I/O programs. This is accomplished by having the SIO logic examine the Jump flip-flop's output when it normally requests the DRT entry from the IOP. A conditional Jump order, with a set Jump flip-flop at the device controller, will cause the jump address to be transferred to the IOP. A conditional jump order and a reset Jump flip-flop will cause the SIO logic to issue a regular request for the DRT contents. Thus, the I/O program can be caused to jump when certain conditions are met in the device controller hardware, or to continue executing sequentially if the conditions are not met. Additionally, an unconditional jump can be performed by setting a bit 4 of the JUMP order to zero. The SIO logic will perform as described above for a successful conditional jump.

Finally, execution of the RETURN RESIDUE instruction does not generate any signals to the device controller, but gates the SIO logic's ORDER RAM to the IOP via the IOP DATA bus.

With the above information it now becomes possible to describe how the SIO logic combines a series of address-data transfers together to perform the necessary control functions to the device controllers.

Each I/O order follows a specified sequence of address-data transfers, first to be fetched, and then executed. The processing of each order can be in one of four states:

(a) Fetch the 1st word of the I/O order

(b) Fetch or Store into the 2nd word of the I/O order

(c) Fetch or Store into the DRT

(d) If READ or WRITE, transfer data until the word count rolls over or the device terminates the order with DEVice END

The <u>next</u> state to be entered by the SIO logic when it services a particular device is kept in the state RAM. The state RAM is updated every time the SIO logic services a device – its update is determined by the state just unloaded from the state RAM and the particular order currently being processed. The state sequences vary for each order, and are presented in figure 20. In figure 20, the circles correspond to the states stored in the state RAM, while the transitions can be viewed as an address-data transfer. The letters assigned to the states correspond to the 4 state descriptions given above. A complete flowchart detailing all the actions taken by the SIO Program Control Logic is given in Figures 31 through 40.

The SIO logic operates identically on all orders in state A by loading the data it receives from the IOP into its order RAM, loading state B into its state RAM, and restoring the incremented address it unloaded from the address RAM.

In state B the logic examines the new order, determining what signals to issue and what next state to load into the state RAM. As can be seen, for READ or WRITE orders, the next state is state D, and for all other orders except END it is state C. No next state is stored if the order is END. When information is stored into the RAMs during the data transfer sequence, the IOP DATA bus will be loaded into the address RAM, the order RAM will be loaded with its old contents, and the state RAM will be loaded with the proper next state.

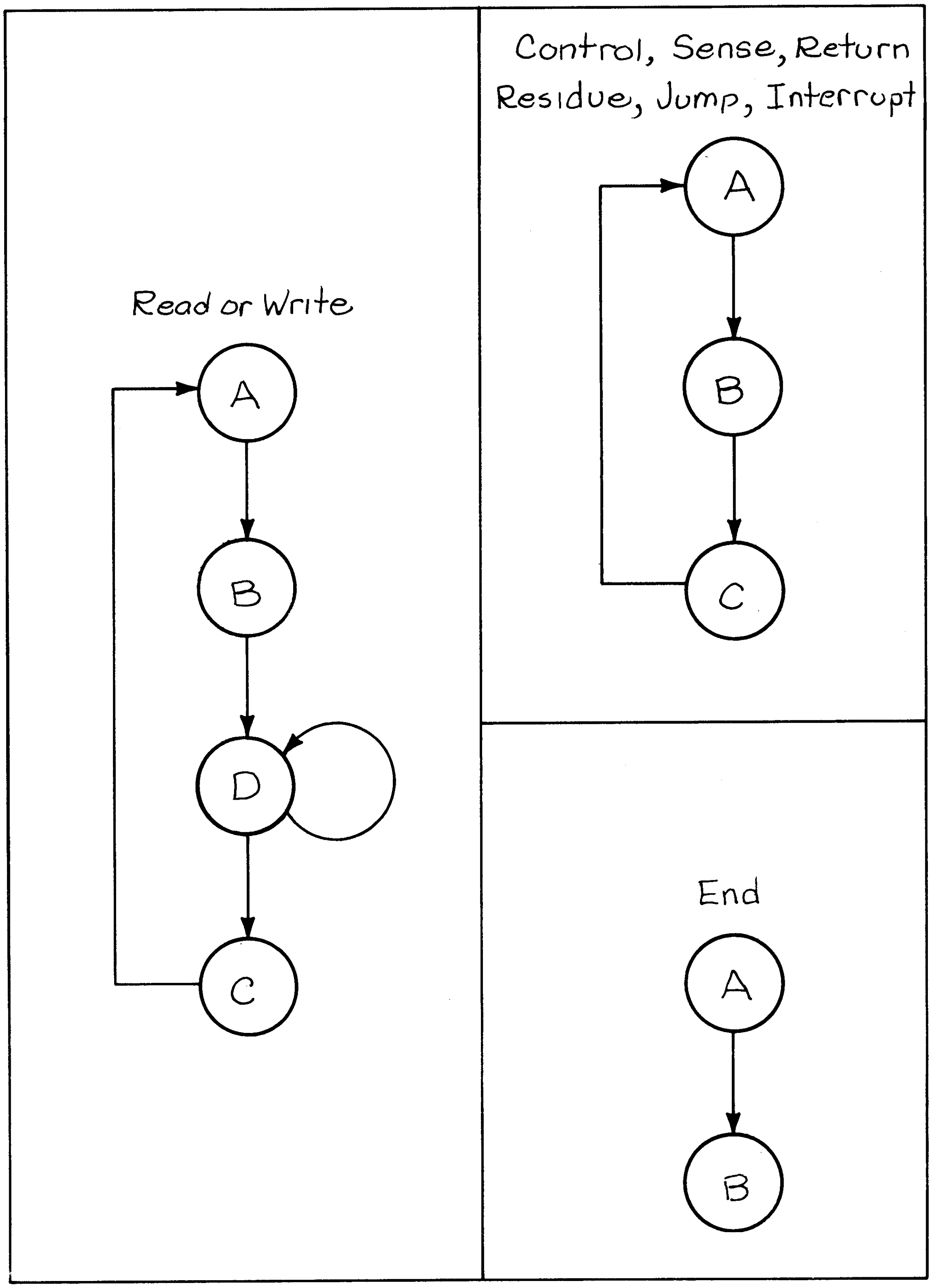


Figure 20. SIO State Transitions

The next state for state C is A. In state C, the SIO logic will always gate the device number to the IOP during the address transfer sequence. All orders but JUMP will cause the returning data to be loaded into the address RAM. An unconditional jump, or a conditional jump with the device controller's Jump flip-flop set, will cause the SIO logic to gate the address unloaded from the address RAM to the IOP. At the same time, the address will be loaded back into the address RAM to be used later as the address of the next order. If a conditional jump is not met – i.e., the device controller's jump flip-flop is not set – the JUMP will proceed as all the other orders by loading the data returned by the IOP into the address RAM.

In state C, in all cases, the order RAM will be loaded with its old contents and the state RAM will be loaded with a next state of A.

The last of the four states, state D, is entered whenever data is to be transferred to or from the device. The next state for state D is state D until the word count rolls over or the device controller terminates the transfer. Both the address and the word count are incremented for every data word transferred to or from the device. They are restored into their respective RAMs and the next state of D is stored into the state RAM. When the word count rolls over, the next state loaded into the state RAM becomes C.

If the device controller terminates the READ or WRITE prematurely, the SIO logic will receive a DEVice END. The receipt of the DEV END will force the SIO logic to execute the functions normally done in state C, and will cause the next state loaded into the state RAM to be state A.

This completes the description of the SIO Program Control Logic. To summarize, it has been shown that the SIO logic executes I/O programs one address-data sequence at a time, interleaving the transfers for various programs. The rate of execution for a single program is dependent on its device controller's service priority relative to all other device controllers running SIO programs at the same time.

Each address – data transfer is comprised of 2 main segments – an address transfer to the IOP and a data transfer. During the address transfer sequence, the RAMs on the SIO card are addressed and unloaded into registers, and in most cases the address RAM contents are sent to the IOP. During the data transfer sequence, the RAMs are reloaded, the source of data being either the registers or the IOP DATA bus. Concurrent with these operations signals are sent to the device controller being serviced.

Each address – data sequence completes one state in the execution of an order, with the state transitions determined by the order being executed. This process continues until the last transfer is completed, namely the device controller status transfer during the execution of an END order.

The following sections will describe the other 2 portions of the SIO Card – the Parity Control Logic and the Diagnostic Control Logic. They will detail the error detecting capabilities of the SIO card and the hardware diagnostic provisions made so that the SIO Program Control Logic can be tested under software control.

7.2 PARITY CONTROL LOGIC

A portion of the logic on the SIO card is used to monitor several parts of the SIO Program Control Logic and the IOP DATA bus during execution of I/O programs.

First, every transfer to the SIO card is checked for odd parity. Secondly, the memory may detect 2 violations: bad parity or non-existent memory being addressed. It would then generate a parity error signal back to the SIO logic. If the parity check by the SIO Logic detects an error, or if the memory sends back a parity error signal, the SIO Parity Control Logic will generate a XFER ERROR signal to the device controller then being serviced.

This signal is to be clocked into the device controller's XFER ERROR flip-flop on the trailing edge of Channel SO. When the flip-flop sets, the device controller should terminate its I/O program by asserting its CLEAR Interface Logic signal, and then generate an interrupt to the CPU. The XFER ERROR flip-flop should also be provided as a status bit.

The Parity Control Logic will also monitor the State RAM by checking for an even number of bits coming from the RAM. Since only one of the four bits should ever be set at a time, the check will catch a faulty RAM or Next State Logic. If this error occurs, the Parity Logic will generate an interrupt of its own to the CPU. This will help to isolate failures to one area of the I/O control chain should they occur.

Every address transfer to the IOP must have an accompanying odd parity bit. Since the parity generators require 90 nanoseconds, worst case, to generate parity, it is generated in advance and stored in an auxiliary RAM. Refer to figure 18. The generators are placed on the data input lines to the RAM and will generate, or check, parity on the data present on that bus.

If the address RAMs are to be loaded from the register, the registers will be enabled onto this bus beginning with the address transfer cycle. The address may be incremented immediately following its transfer, so that correct parity will be generated in the worst case, 140 nanoseconds after the trailing edge of the data poll. This parity bit will be written into the auxiliary RAM with Service Out, which can be issued no sooner than 160 nanoseconds after the data poll is removed. Thus sufficient time is available to update the address and its parity before they are loaded back into the RAMS.

When the address RAMs are loaded from the DATA BUS, the parity bit that is stored is the bit sent by the IOP. This is due again to the long worst case time to generate the parity locally. The generators will be used, however, and the parity they produce is compared to the parity bit sent. If the 2 differ, an XFER ERROR will be issued to the selected device controller to terminate its program.

Lastly, the parity bit is unloaded from the auxiliary RAM every time the RAMs are accessed during an address transfer. Therefore, the parity bit for the address is available simultaneously with the address, thus avoiding the problem of delaying the parity for each address by the generation time.

7.3 DIAGNOSTIC CONTROL LOGIC

The SIO Diagnostic Control Logic has been included on the Multiplexed SIO card to allow checkout of the SIO Program and Parity Control Logic. As indicated earlier in this manual, the Diagnostic Logic will allow complete diagnostics for all but the control signals to the device controllers.

The formats for the diagnostic command and status words are given in Figure 21, along with the format of the information returned with the state RAM output. What follows is a description of the functions of each of the bits in the three words.

The select number in the Control Word is used as the RAM address. Each number corresponds to the same number service request line from the device controllers. Thus, the portion of the RAMs used by a particular device controller can be examined by issuing a control word with that controller's select number.

Bits 4, 5, and 6 of the control word indicate which set of RAMs of the given select number is to be operated upon. Only one of these 3 bits should be set at a time.

Bit 7, the Load bit, informs the Diagnostic Logic that the registers should be loaded from the RAMs before the registers are gated to the IOP DATA bus during a READ.

When bit 8 is set, either the address register, or the order register will be incremented after it is read. The register incremented is determined by which of the two bits, the address bit or the order bit, has been set.

The sequence of events to perform a diagnostic operation is to issue a control word followed by any number of READS or WRITES. Each RAM can be checked by writing any bit pattern into it and then reading it back through the registers to the CPU.

The counting function of the registers can be checked by issuing the following sequence of commands from the CPU:

1. Control with the Load and Incr bits set
2. Write
3. Read
4. Control with the Load bit reset, and Incr bit set

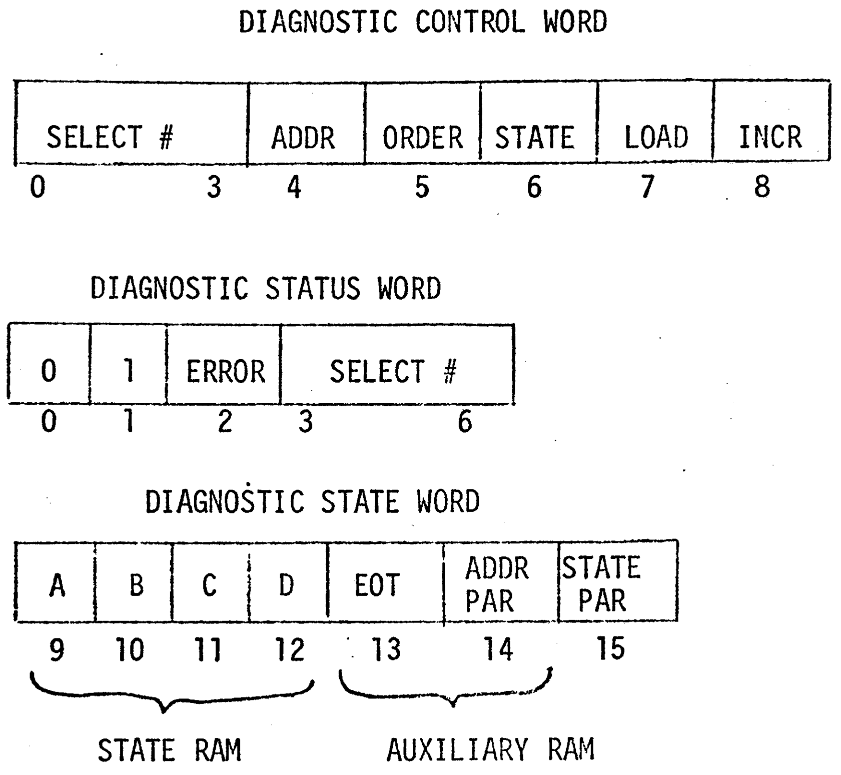


Figure 21. SIO Diagnostic Word Formats

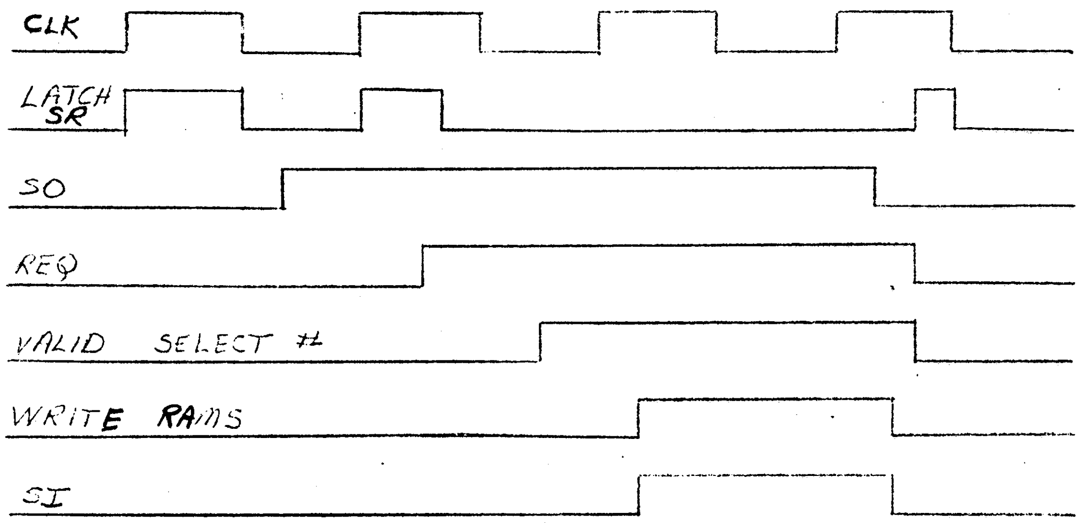


Figure 22. Initiation of an SIO Program

Successive Reads should then increment the register, allowing the counting function to be checked.

When the order register is incremented and rolls over, it will set the EOT FF. The EOT FF will remain set until the state B is unloaded from the state RAM, and it will be stored in the auxiliary RAM whenever a WRITE command is issued.

The diagnostic state word is used to verify 2 portions of the program Control Logic—the next state function and the auxiliary RAM inputs.

Reading the STATE RAM will return the contents of the corresponding auxiliary RAM address, and parity on the state read from the state RAM. Besides the EOT FF bit, the auxiliary RAM stores the address parity bit and the data chaining bit of the order register. This last bit is used by the SIO Program Control Logic and is not of concern during diagnostic operation. During diagnostics, these 2 bits will also be written into the auxiliary RAM with the EOT bit whenever a write command is issued. It should be noted that the address parity bit will only be correct during diagnostics when the address RAM is loaded from the address register.

This occurs because data sent from the CPU does not have parity, but whenever the address RAM is loaded from the DATA BUS, the parity bit is loaded from the parity line.

To check the next state logic the following command sequence should be followed:

1. Control with order bit set
2. Write of the desired order
3. Control with state bit set and order bit reset
4. Read of the state
5. Write of desired starting state

If the above sequence is followed by successive Reads and Writes of the state RAM, the next state function for the order set in the order RAM can be verified. Each write to the state RAM must be with 0 for data. The data then written will be determined by the next state logic and can be read with the succeeding Read State Command.

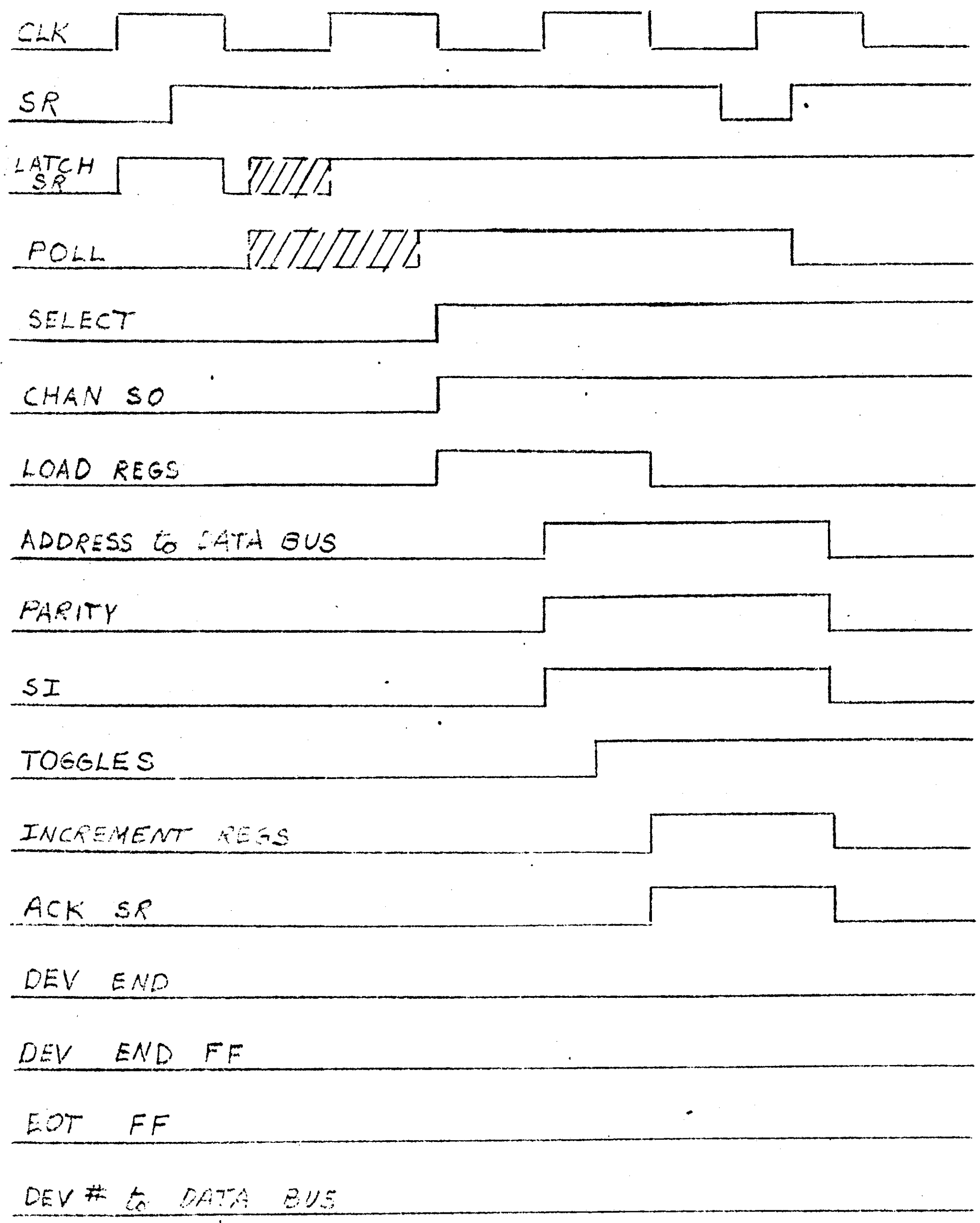


Figure 23. Address Transfer Timing

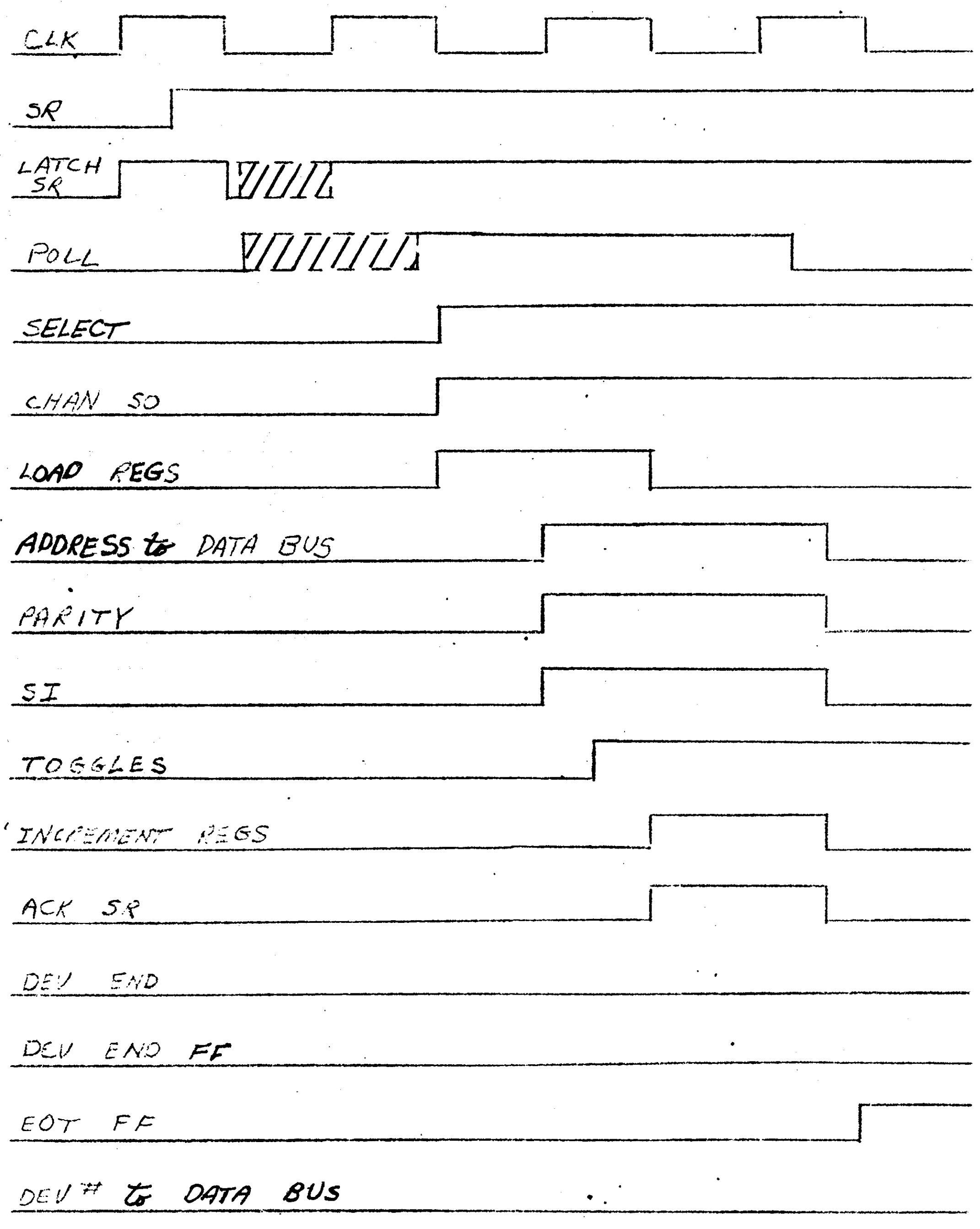


Figure 24. Address Transfer with Word Count Rollover

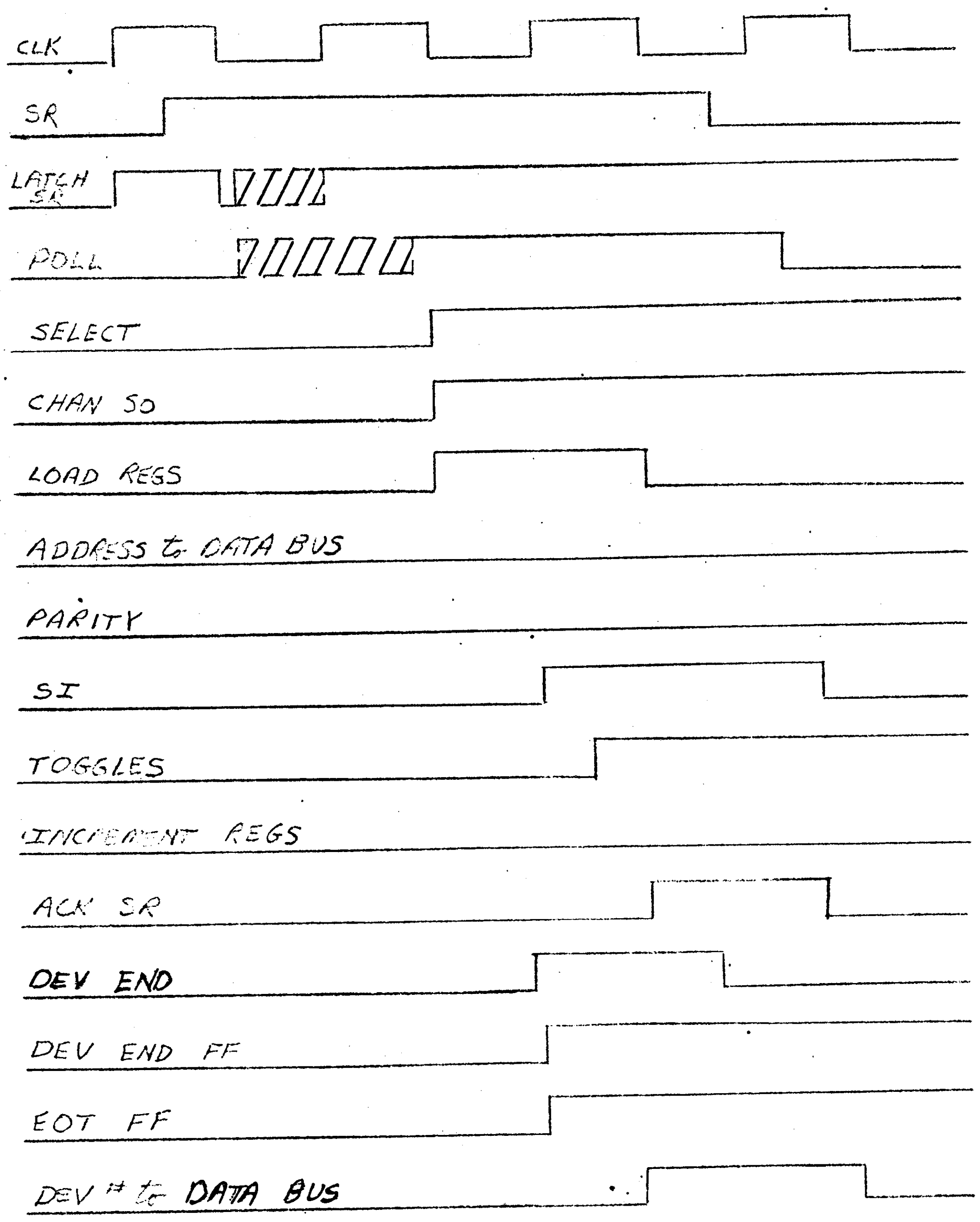


Figure 25. Device Number Transfer Due to Device End

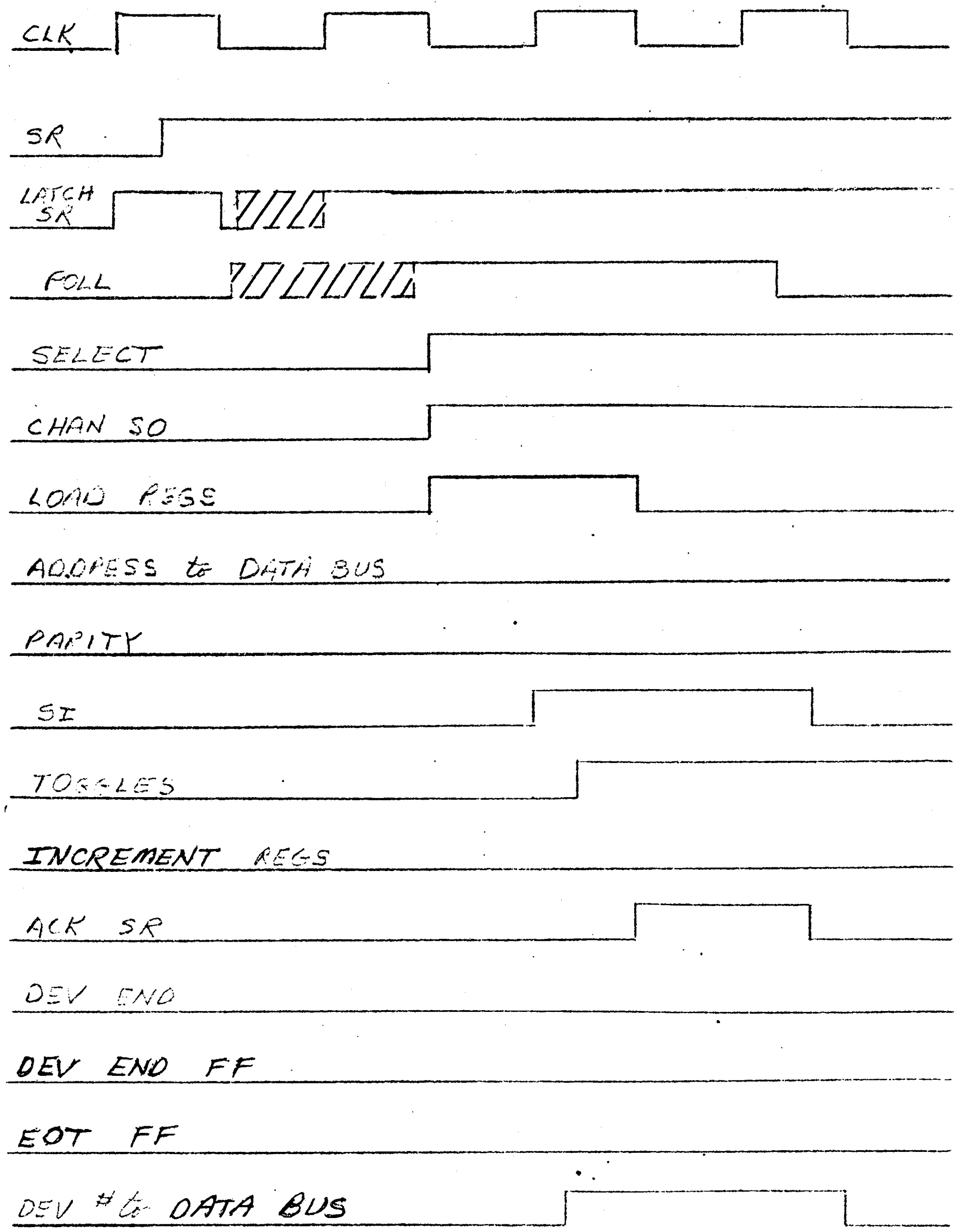


Figure 26. Device Number Transfer, Without DEV END or WC = 0

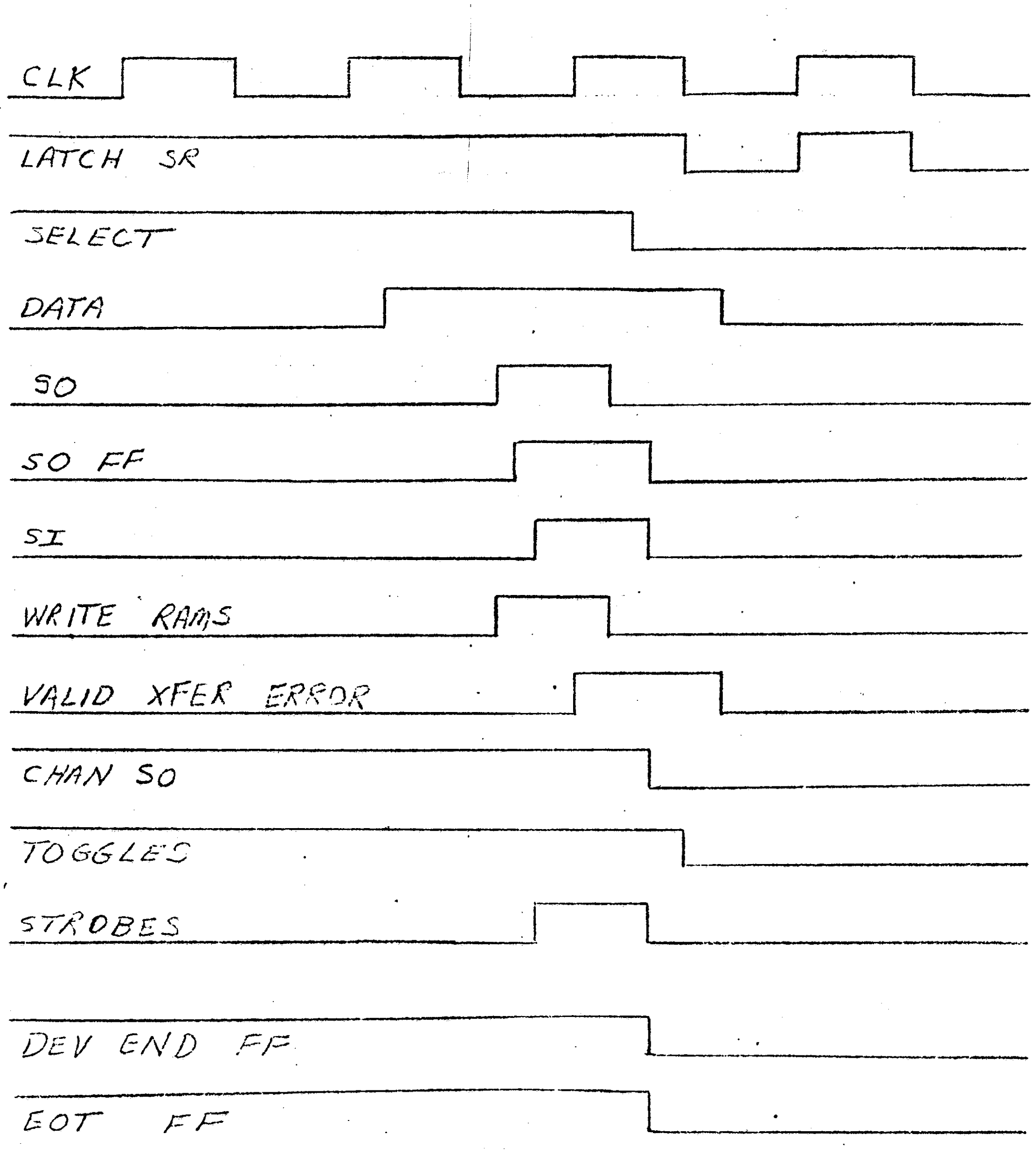


Figure 27. Outbound Transfer with Transfer Error

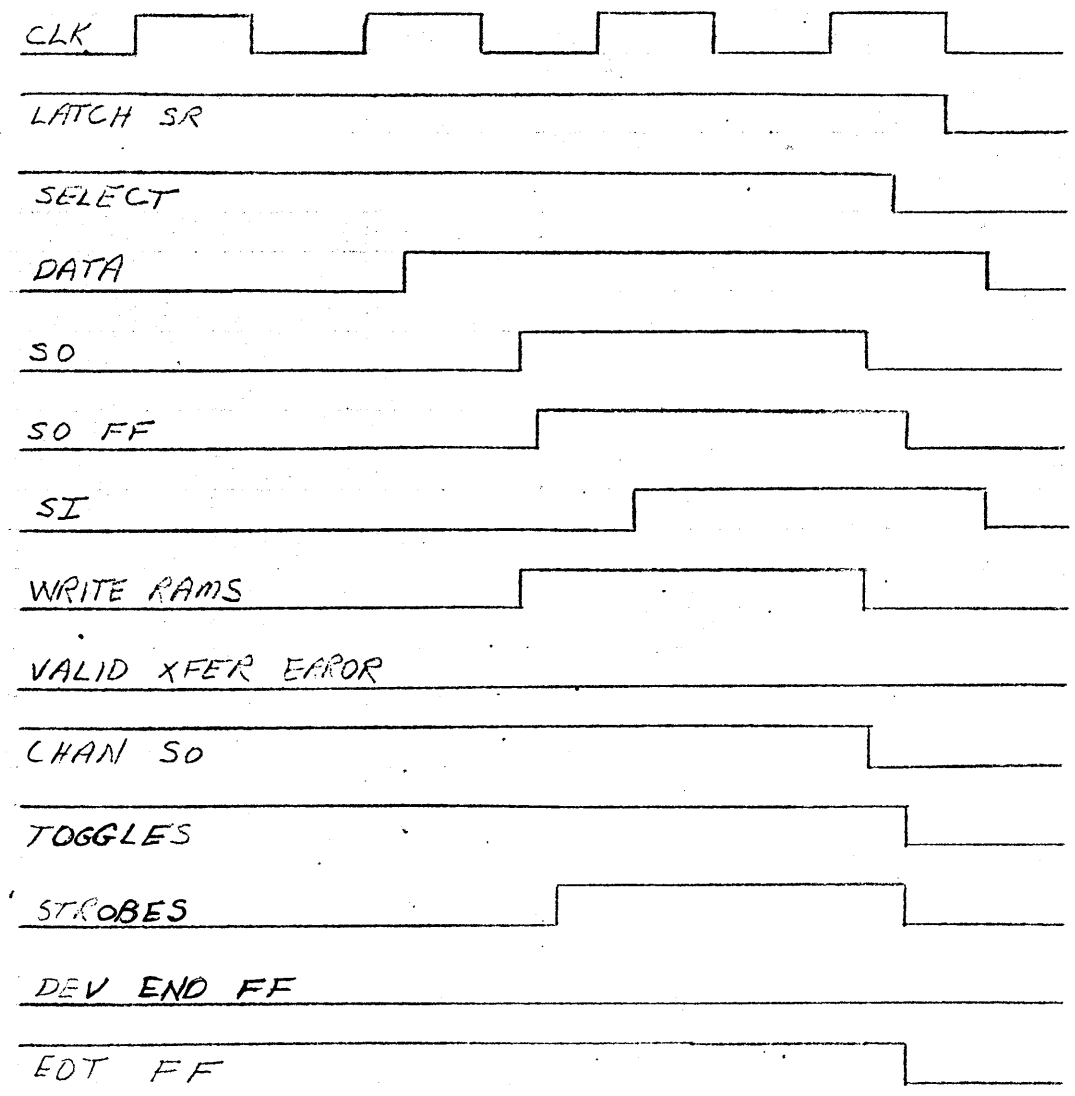


Figure 28. Write to Controller, with WC = 0

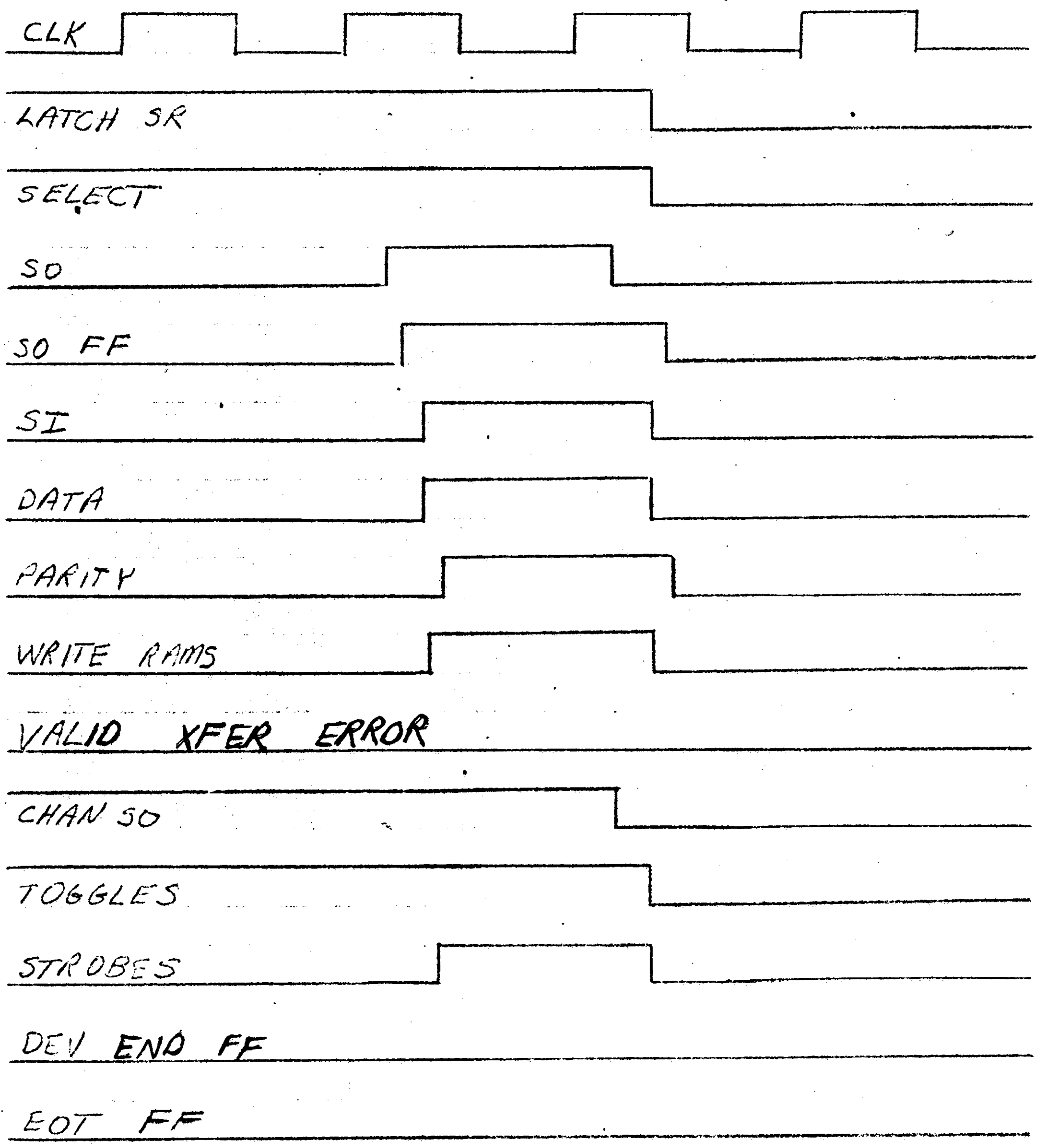


Figure 29. Return Residue

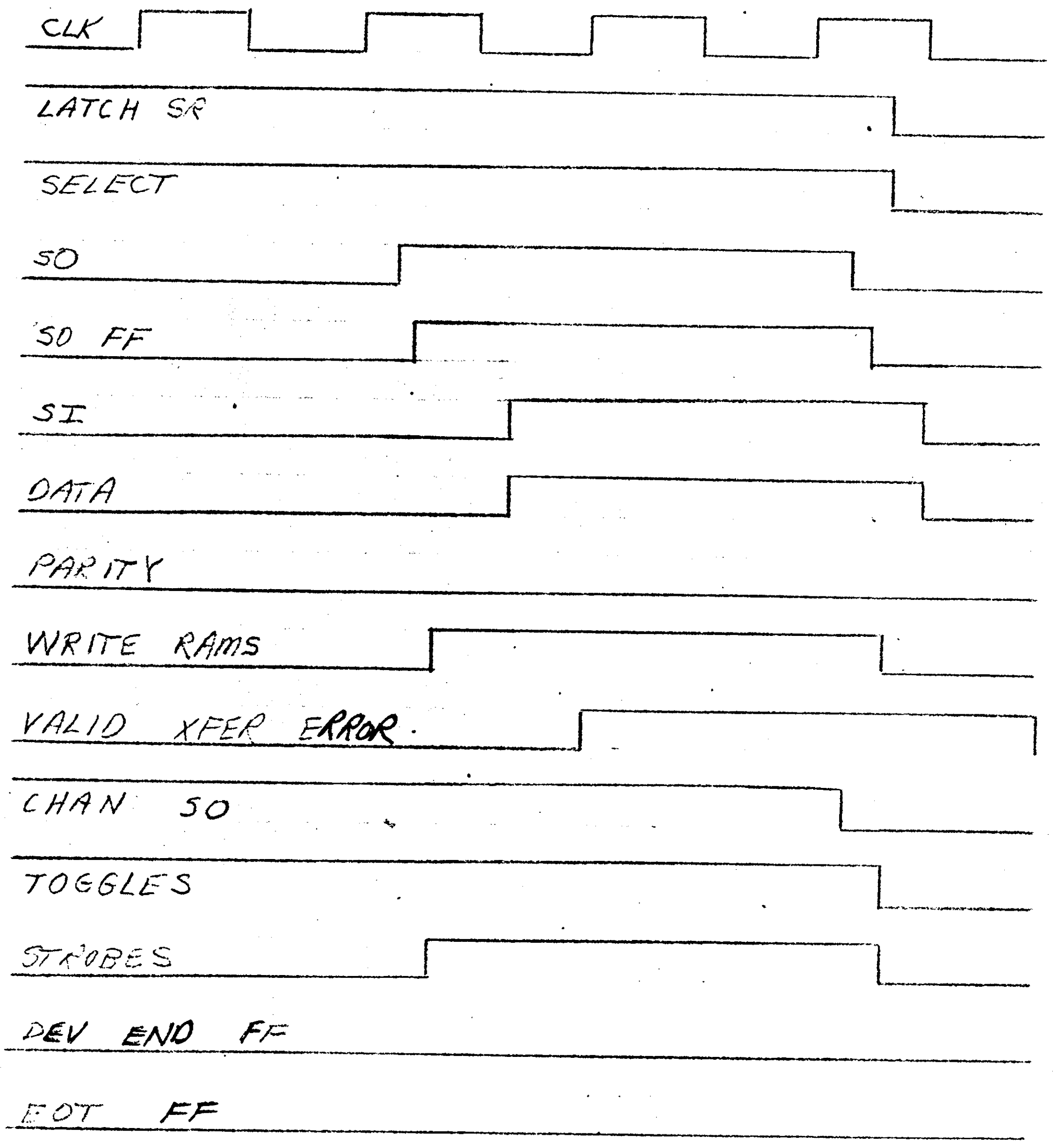


Figure 30. Inbound Transfer, Error Detected by Memory

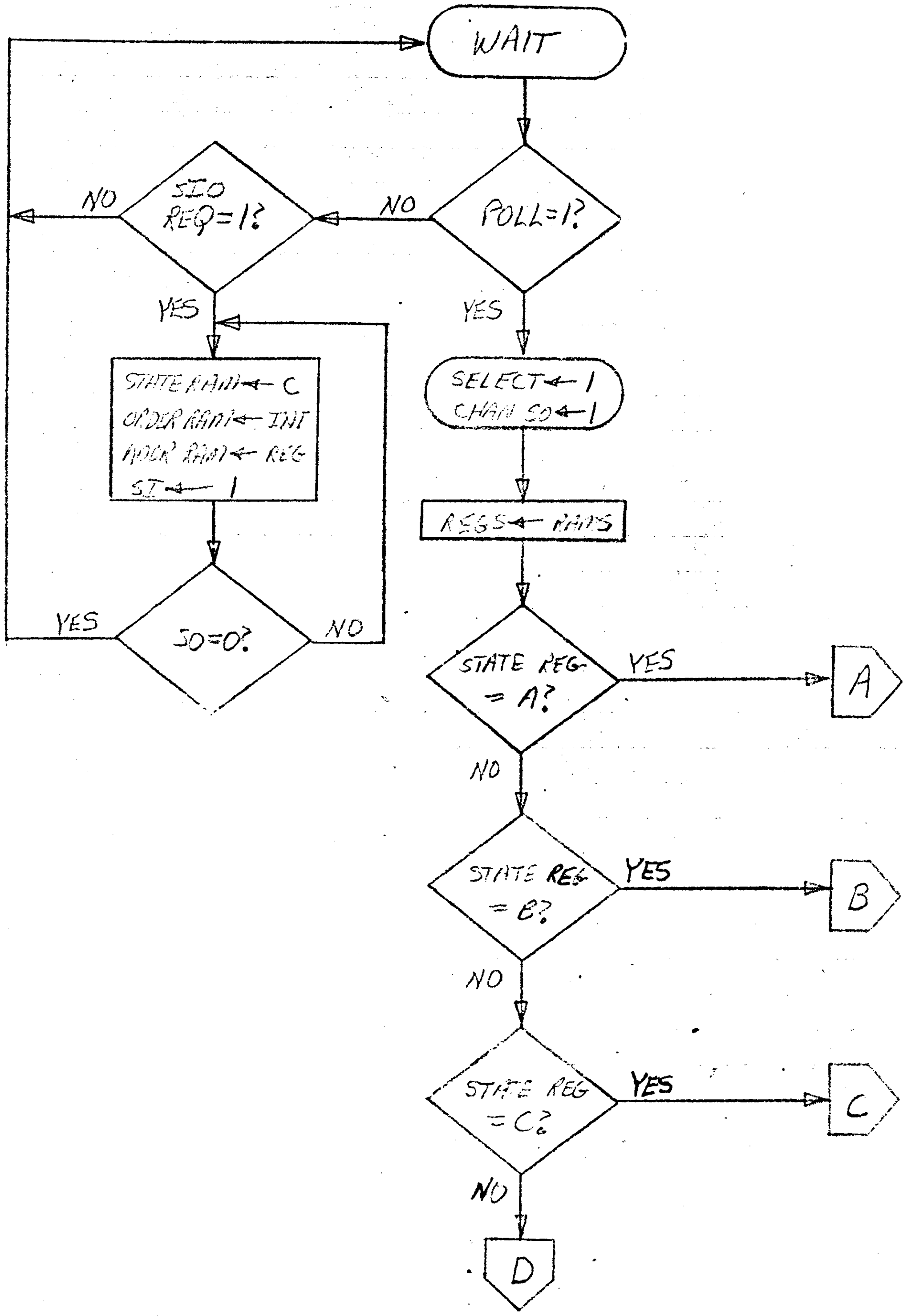


Figure 31. SIO State Branches

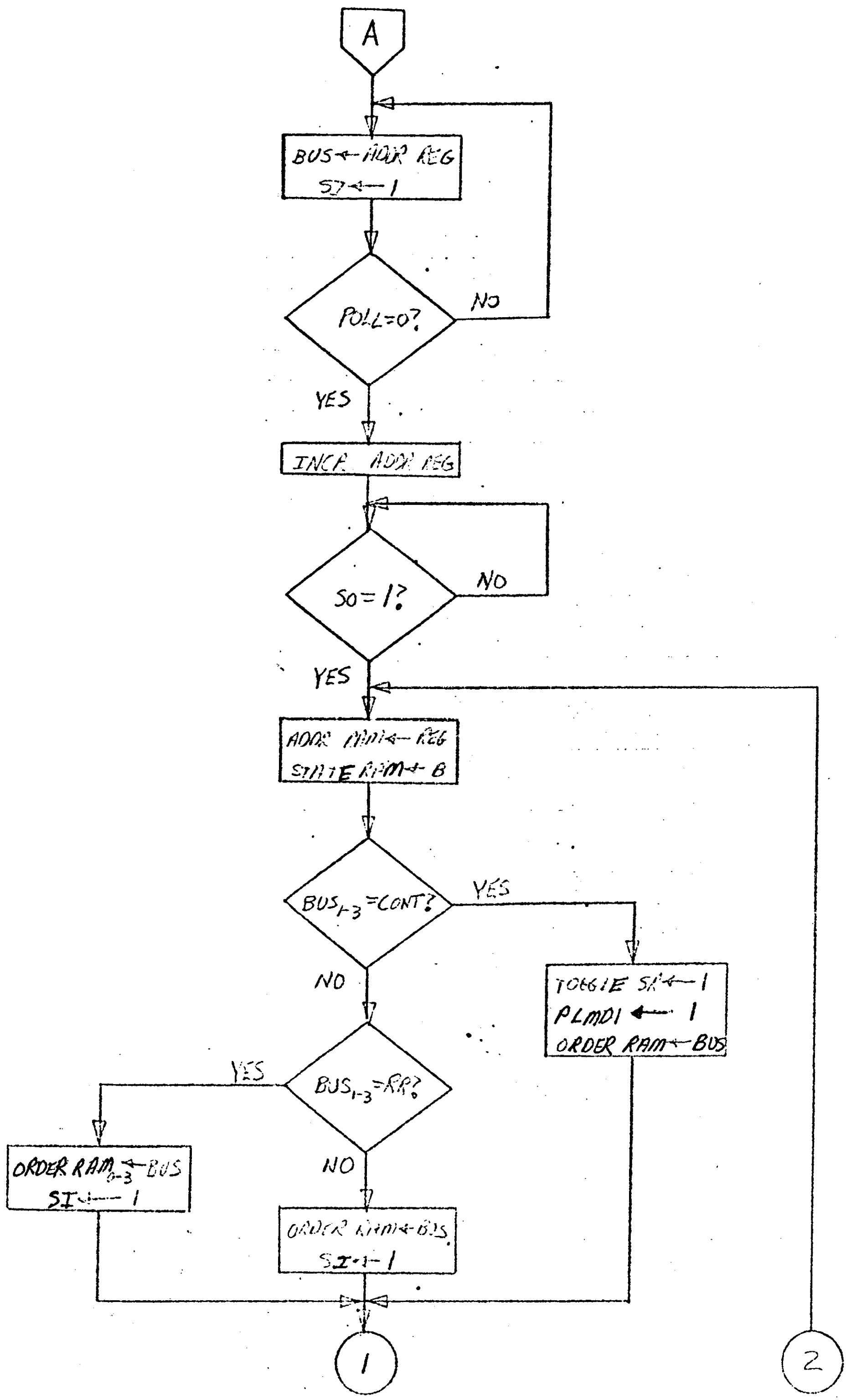


Figure 32. State A, Part 1

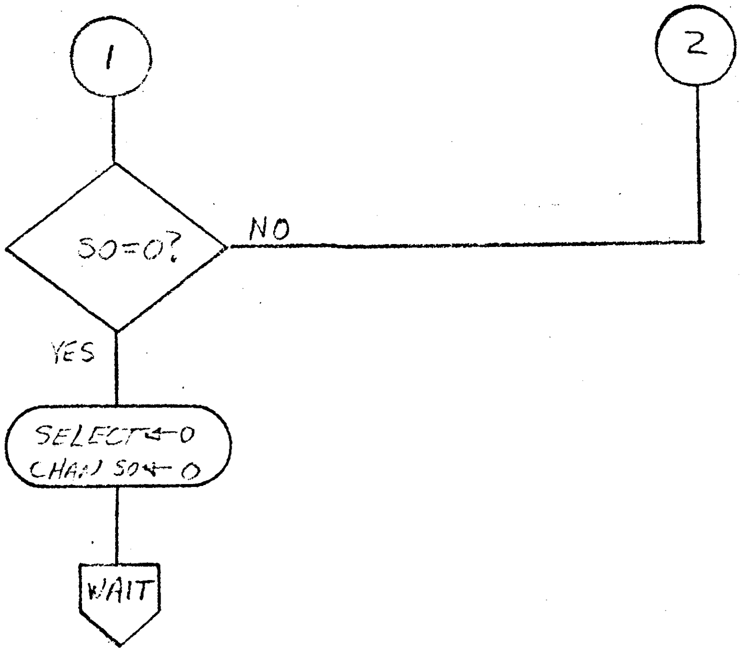


Figure 33. State A, Part 2

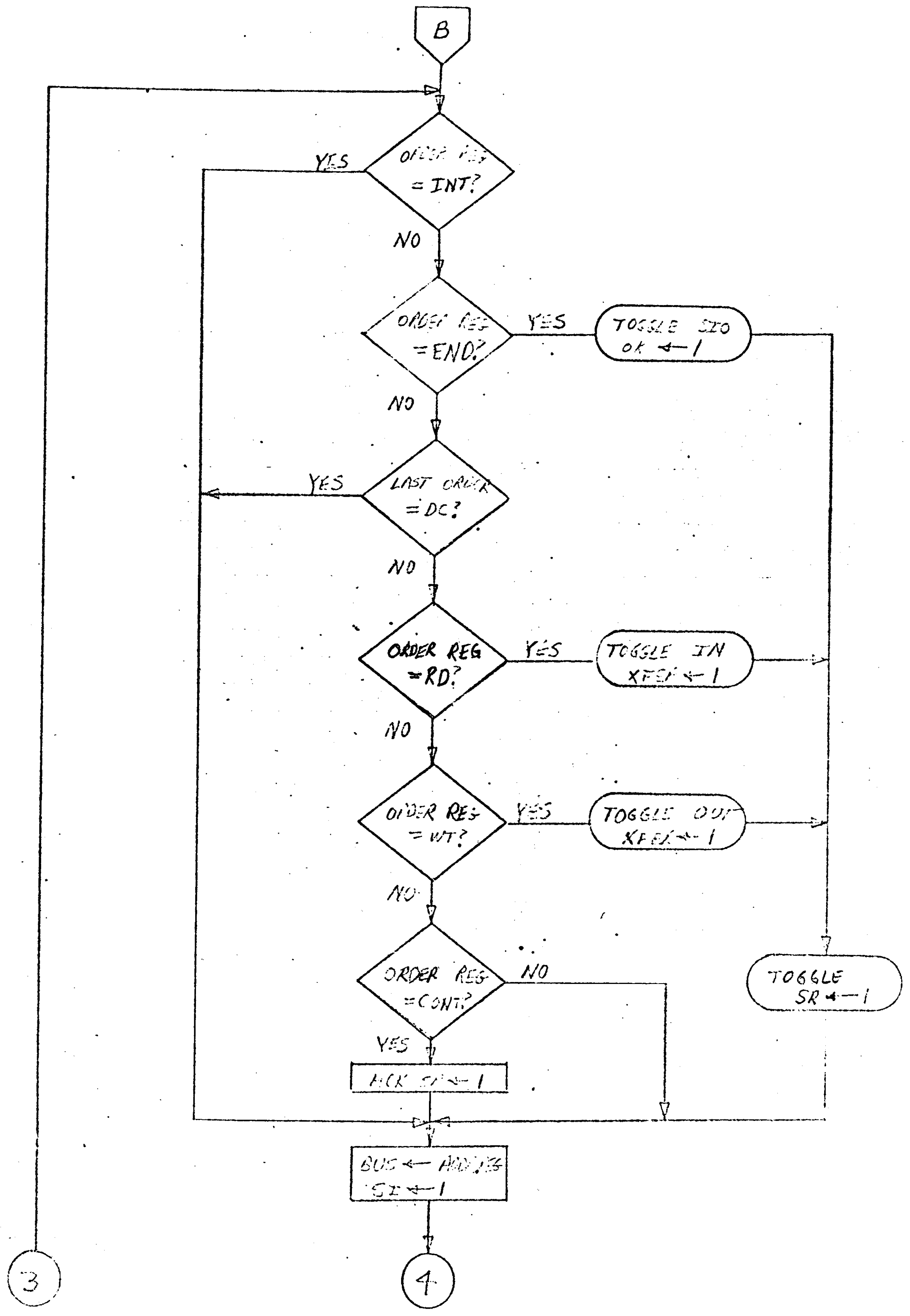


Figure 34. State B, Part 1

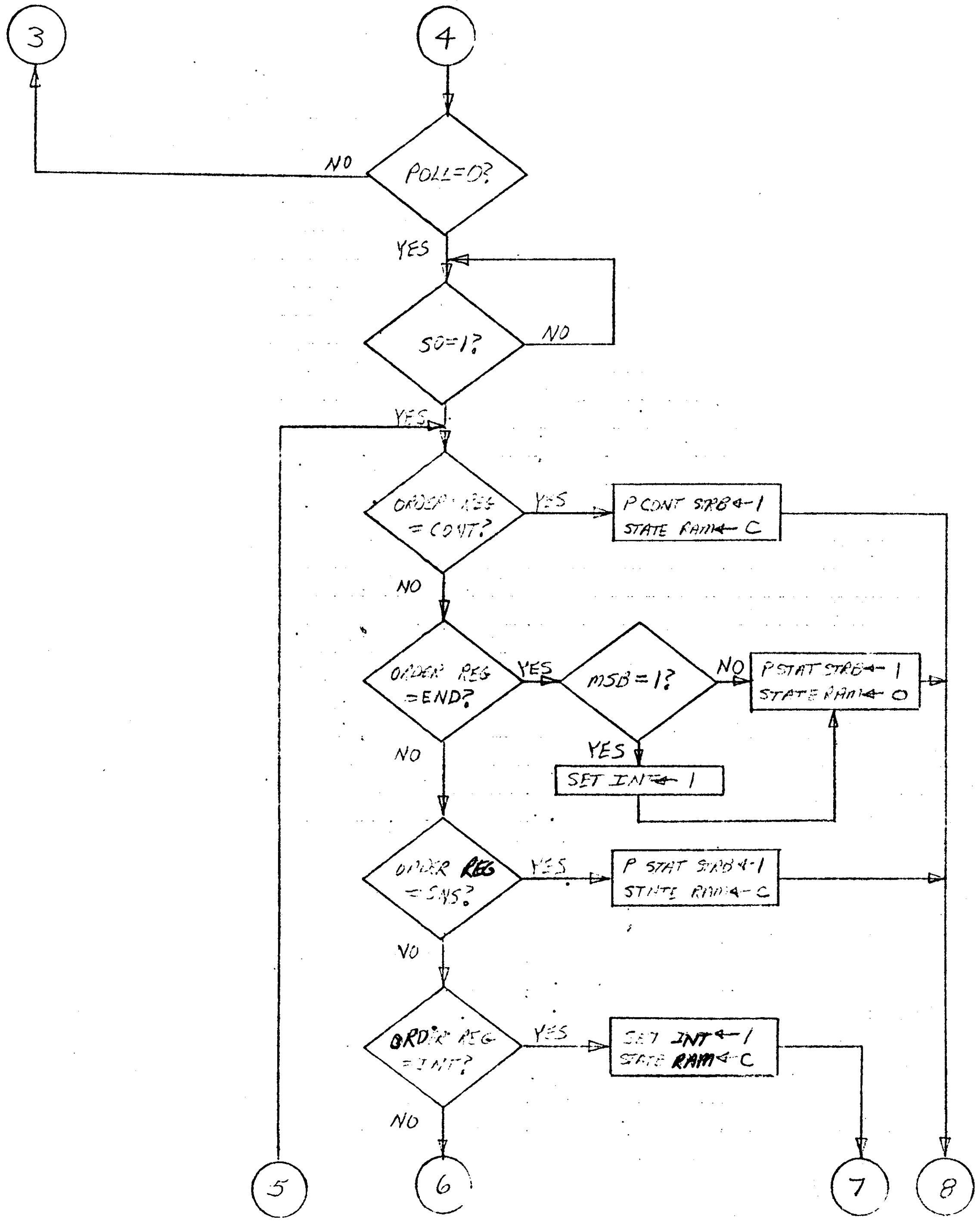


Figure 35. State B, Part 2

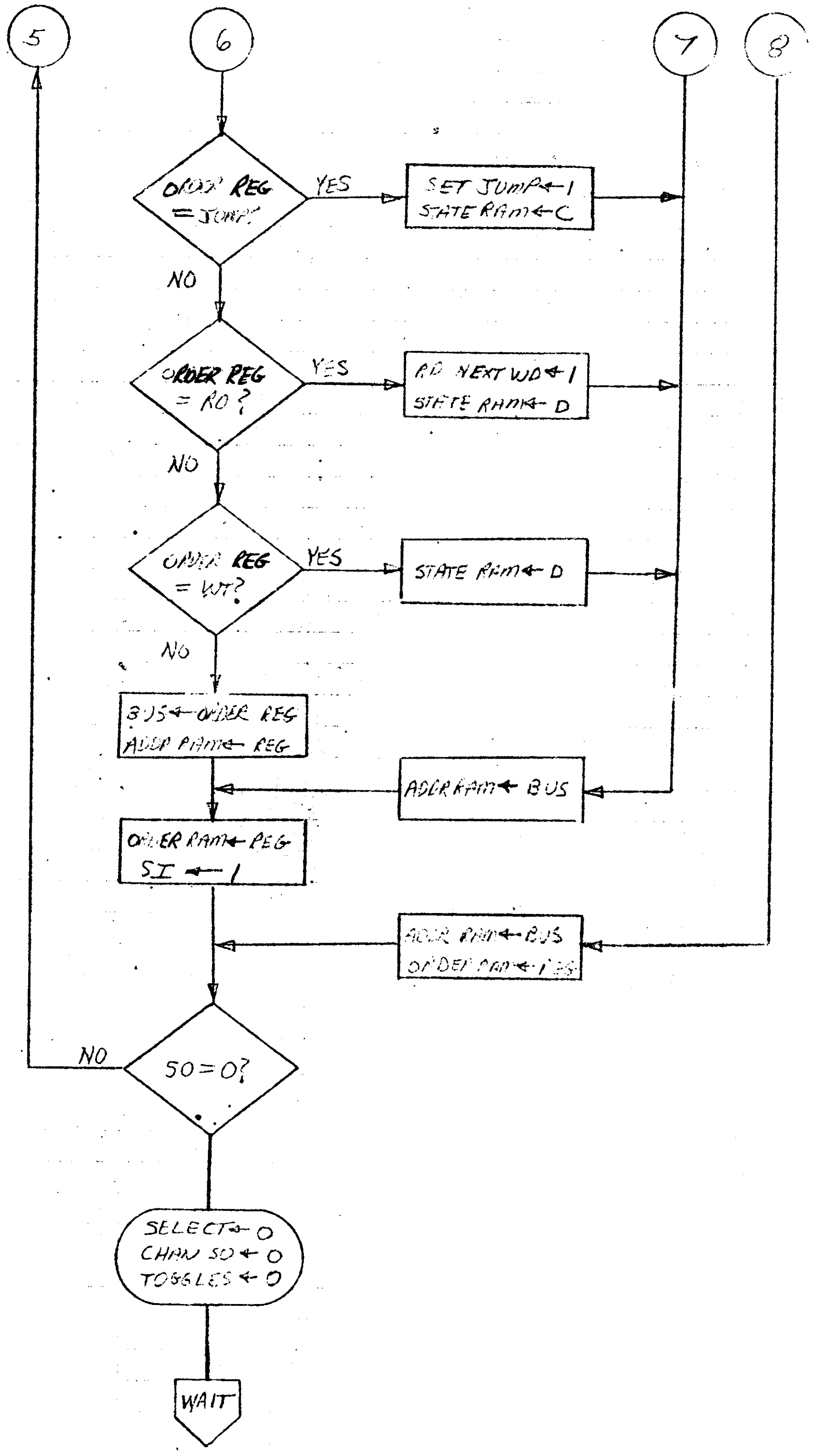


Figure 36. State B, Part 3

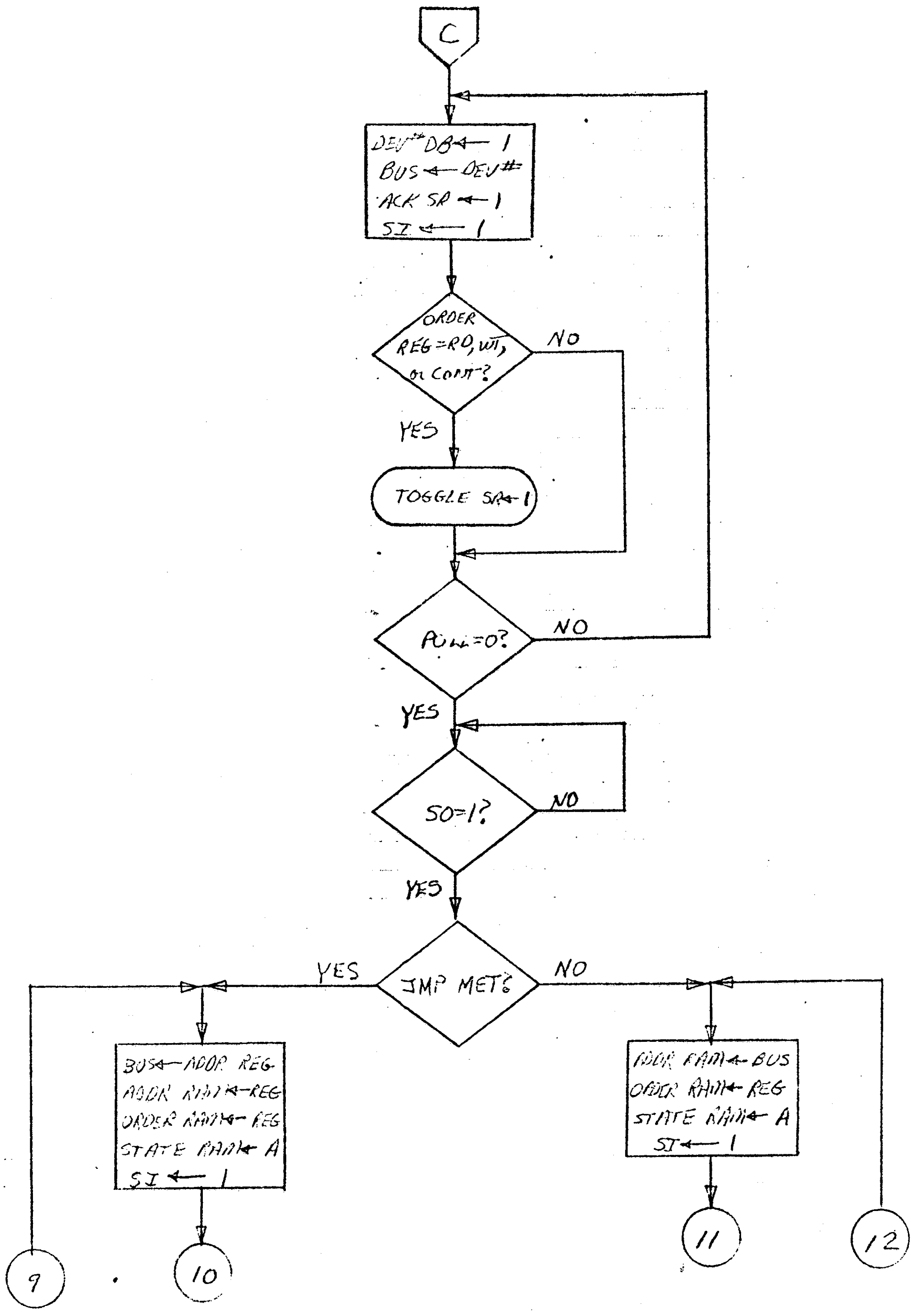


Figure 37. State C, Part 1

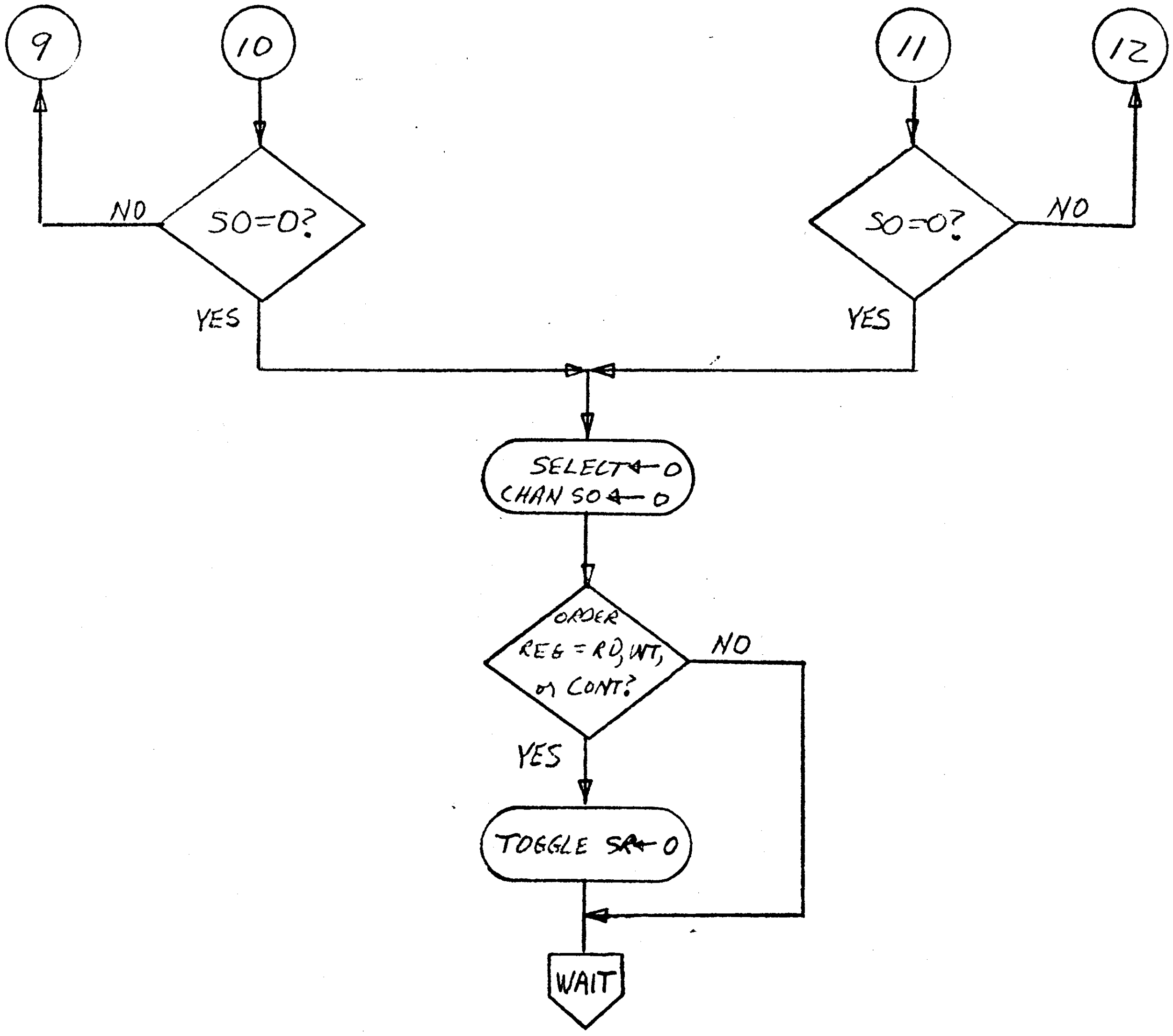


Figure 38. State C, Part 2

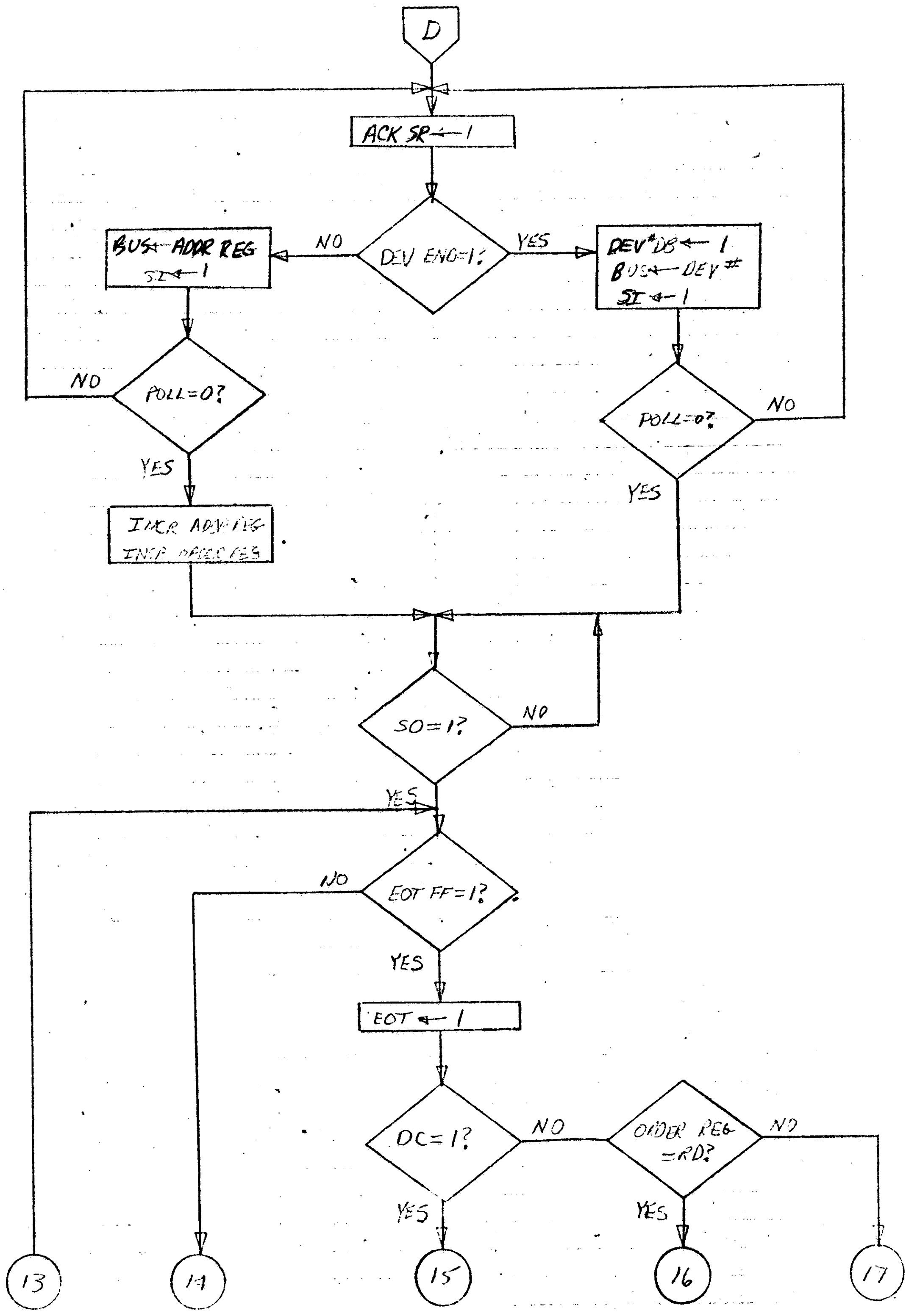


Figure 39. State D, Part 1

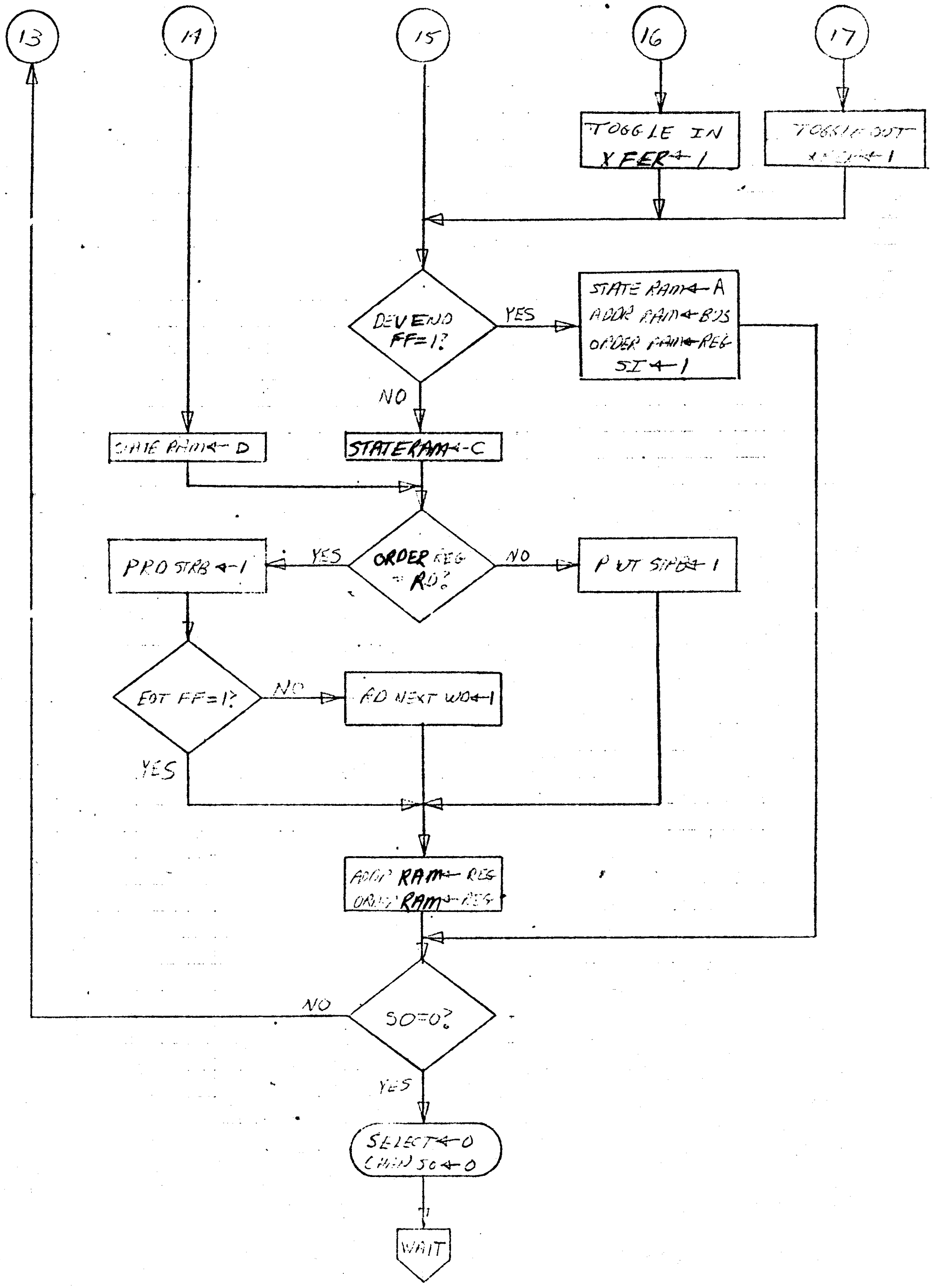


Figure 40. State D, Part 2

If the last address in the State RAM is known, the address parity circuit can be checked when the state word is read back to the computer by checking the address parity bit of the state word.

By setting the EOT FF as explained earlier, the next state function for a READ or WRITE order can be fully checked. In both cases, the State RAM should be set to state D. the data transfer state, since the EOT FF must be set to exit state D. The set EOT FF can be verified by checking the EOT bit in the state word when it is read into the CPU.

The diagnostic state word, then, contains the information needed to check the parity generation logic and the next state function of the Program Control Logic.

The status word of the SIO card has 7 valid bits. Bit 0, the SIO permit bit, is always false, while the READ-WRITE permit bit, bit 1, is always true. Bits 3-6 contain the select number of the device controller last serviced by the SIO card, and bit 2 will be set if the state RAM parity for that select number was found to be in error.

In summary, the Diagnostic Control Logic permits verification of the RAMS, the counters, all data paths, the next state function of the Program Control Logic, and much of the Parity Control Logic in a stand alone mode. If all of the above is deemed operable by the diagnostic program, the remaining logic can be verified as operable by issuing an SIO program to a selected device controller and then examining the results of that program for any errors.

7.4 TRANSFER SEQUENCES

7.4.1 Address Transfer Sequence

Refer to figure 18 for the following discussion. Each of the 16 device controllers controlled by the SIO card has a separate service request line. Every Service Request generated by the device controller is acknowledged by the ACK SR line. This signal will reset the controller's service request flip-flop when the request is honored by the SIO card. The signal is issued 1-1/2 clocks after the SIO card receives the data poll and it will last approximately 80 ns.

Service Requests from the device controller are required:

1. For every transfer during a READ or WRITE

2. To continue program execution after a READ or WRITE

3. To continue program execution after each of the 2 command transfers of a CONTROL order.

Since the above cases are the only times service requests are required from the device controller, they are the only times ACK SR will be issued. Thus asynchronous setting of service requests is possible, as the requests will not be acknowledged until they are actually needed by the SIO logic.

The state of the Service Request lines is latched up during each positive half of the system clock cycle and during the entire duration of an SIO address – data transfer. If a data poll is received by the SIO logic during the positive half of the cycle, the Service Request latches will remain latched for the entire SIO transfer, otherwise the latches are freed to pass a new state of the service request lines onto the priority encoders during the second half of the clock cycle.

The priority encoders are used to generate a 4 bit RAM address and a device select code from the highest priority service request present at their inputs, when the SIO card is not in the diagnostic mode. During the running of diagnostics the priority encoders are disabled and the RAM address is obtained from latches loaded under CPU control.

Under normal operation, if a data poll is received during the positive half of the clock, the select code determined by the encoders is returned to the device controllers, along with a strobe line called Channel Service Out.

The first falling clock edge after receipt of the poll will start the loading of the registers from the RAMs. The load signal will last for one full clock cycle, terminating on the next falling edge. Worst case delays indicate that valid data will appear at the register outputs no later than 80 nanoseconds after the load pulse begins. Thus the control logic will have valid information from the RAM's one clock cycle after a poll is issued.

At the same time the RAMs are being unloaded the select code is sent to the device controllers. That device controller with the select code sent by the SIO card will enable 16 SIO command and response lines. If the device controller is executing a read or write order and terminates the transfer prematurely, it will issue a DEV END signal to the SIO logic upon receipt of its select code from the SIO logic. Worst case gate and transmission delays indicate the DEV END signal will reach the SIO logic about 80 nanoseconds after the select code is issued by the SIO card. Thus the DEV END signal and valid information from the RAMs will reach the SIO control logic simultaneously, in the worst case, one clock cycle after a data poll is issued.

If a DEV END was issued or the SIO logic determines it has finished the last I/O order, it will issue the control signal DEV # to DATA BUS, which will place the selected device controller's device number onto the IOP DATA bus. At the same time the SIO logic will generate the handshake response to the data poll, Service In. In a worst case situation, the SI signal will be returned after one clock cycle from the issuance of the data poll, while gate and transmission delays will delay the strobing of the device number onto the data bus for an additional 80 nanoseconds, or 1/2 clock cycle. This gives a worst case data bus settling time of 80 nanoseconds before the IOP strobes the data into its address register.

If a DEV END is not issued and the SIO logic is executing an order, it will gate its address register onto the data bus at the same time it generates SI, or one clock after the data poll was issued. This affords a data settling time of 160 nanoseconds for all Address Transfers from the SIO card.

In either of the above cases, the SIO logic will return a CMD IN Signal to the IOP with the SI. The four commands to the IOP are:

1. DRT transfer
2. Jump in progress
3. Transfer to memory
4. Transfer from memory

In actuality three signal lines are sent to the IOP, one for each of the first three commands. The fourth command, transfer from memory, is assumed if none of the other three signals is asserted.

When the IOP receives the SI and CMD IN signals it will strobe the data bus into its address register on the next rising system clock edge, providing the SI was received during the previous high portion of the clock cycle. This will provide a minimum of 80 nanoseconds deskewing on data received by the IOP from the device controllers.

When the data is strobed by the IOP the data poll is removed. This will cause the SI and IOP DATA bus signals to be terminated, completing the address transfer sequence.

Two points should be noted. First the CMD IN to the IOP remains valid until the data transfer sequence is completed. Secondly, the timing as described above dictates that every address transfer will be 2 system clock cycles, or 320 nanoseconds, in duration.

7.4.2 Data Transfer Sequence

The Data Transfer Sequence is governed by the Service Out signal issued by the IOP. If data is sent from the IOP, it will be present on the IOP DATA bus 1/2 clock cycle before and after Service Out is issued. When data is sent to the IOP, Service Out will be will be used to gate information from the device controller, or SIO logic, onto the IOP DATA bus. The IOP will allow at least 80 nanoseconds for the bus to settle before strobing the data into its register on a rising clock edge. This settling time is achieved in the IOP by latching the Service In line during the 2nd half of each clock cycle. Thus, the Service In signal will be passed through the latch during the positive 1/2 of the system clock cycle and must become true sometime during the positive half of the cycle in order to strobe the register on the next rising edge. This same mechanism for deskewing incoming data is used on the data poll-service in handshake.

Each service out sent by the IOP during I/O program execution will be used in the SIO control logic as the enabling signal for most of the SIO command strobes. The Service In response to Service Out will be generated by the selected device controller whenever one of the commands below is issued to it by the SIO logic.

1. Programmed Read Strobe
2. Programmed Write Strobe
3. Programmed Status Strobe
4. Program Command 1
5. Programmed Control Strobe

In all other cases the SIO logic itself will return the Service In to the IOP.

The Service Out signal will also be used to write information back into the RAM location that was unloaded during the address transfer sequence. The source of the information for each set of RAMs is determined by the control logic. For the ADDRESS and ORDER RAMs, the source is either the registers loaded from the RAMs or the IOP DATA bus. The RAMs will invert data input to them and must be presented with ground true data. Since the IOP DATA bus is ground true, it is gated to the RAM inputs through non-inverting receivers. The data setup and hold times for the RAMs can be met when loading from the IOP DATA bus since the data is valid on the bus 80 nanoseconds on either side of Service Out. When the RAMs are loaded from the registers, the data is gated through an inverting chip, so that the RAMs are again loaded with ground true information. Their outputs will then always present positive true data.

The state RAMs can be loaded from 2 sources – the output of the next state logic or bits 0-3 of the data bus. The second source is used only during diagnostic checkout and is provided to set various states into the state RAM.

Several major actions occur in the SIO logic on the trailing edge of Service Out. The 4 bits of the order register containing the order are zeroed out, as is the state register. This is done to prevent execution of erroneous information while the RAMs are unloaded to service the next request. The trailing edge of Service Out will disable the select code and Channel SO Strobe to the device controller. The falling edge of the strobe will clock four flip-flops on the device controller card. The inputs to these flip-flops are:

1. Toggle Service Request
2. Toggle SIO OK
3. Toggle In XFER
4. Toggle Out XFER

As their names imply, these lines will toggle their respective flip-flops when they are asserted during the falling edge of the Channel So Strobe.

The Service Request flip-flop is used to provide service requests not required from the device controller, but necessary to execute the I/O program. The device controller will issue requests only during a READ or WRITE, to transfer data; after receiving Program Command 1 or Programmed Control Strobe, to allow the controller to act on the control information sent; and after completion of a READ or WRITE, to allow the controller to perform any housekeeping before continuing the I/O program.

The Service Requests needed to fetch all orders and the DRT entry are provided from the Service Request flip-flop. In addition, it will provide the Service Requests necessary to execute the SENSE, RETURN RESIDUE, JUMP, INTERRUPT, and END orders. At the start of each I/O program, this flip-flop will be set. It will be reset whenever execution of a READ, WRITE, or CONTROL begins and set again upon completion of these orders.

The SIO PERMIT flip-flop is used with controller conditions to form bit 0 of the controller's status. This flip-flop will be set at the beginning of the execution of an I/O program. It will be reset when the SIO logic executes the END order and generates Toggle SIO PERMIT during transfer of the device controller status. As long as it is set, bit 0 of the controller's status will be false, causing rejection of any other SIO commands issued by the CPU to that device.

The IN XFER flip-flop is set for the duration of execution of a READ order. If several READ orders are chained together, it will remain set until the last READ is completed. The Toggle IN XFER signal will be sent to the controller during the last data transfer, so the IN XFER flip-flop will be reset on the trailing edge of the last Programmed Read Strobe.

The OUT XFER flip-flop operates identically to the IN XFER flip-flop but for WRITE orders. These two flip-flops are provided to inform the controllers of the initiation and completion of a data block transfer.

One additional signal is provided to the device controller to inform it of the status of its I/O program. This is the End of Transfer signal, which is asserted every time the word count rolls over, or when the device generates DEVice END. It is currently used by the disc controllers to enable them to handle chained READS or WRITES that are not exactly multiples of their sector size.

During data transfers the trailing edge of the data poll increments the word count portion of the order register. When this word count rolls over, it will set an EOT flip-flop in the SIO logic, generating the EOT signal to the device controller. Therefore, the signal will be asserted slightly after completion of the address transfer sequence and will remain valid until the data transfer sequence is finished.

The main actions of the data transfer sequence, then, are to store updated information into the RAMs, set the correct state of the 4 flip-flops in the device controller, and transfer data from either the SIO card or the device controller.

SECTION 8

MODULE CONTROL UNIT

8.1 INTRODUCTION

Module Control Unit (MCU) is the means by which a module interfaces to the MCU bus. The MCU bus is the universal means of communication between all the modules in the ALPHA system. Each module, in order to communicate with another module in the system, must obtain a time slice on the MCU bus and transmit its "message" to another module during that time slice. Each time slice on the bus consists of one clock cycle. Modules, in order to obtain a bus cycle must request the bus from their MCU. Each module has a priority rank such that if more than one module request the bus simultaneously, the module with the highest priority gets the next bus cycle.

Each module in the ALPHA system has an assigned module number. This number is used for addressing the module in all communications with that module.

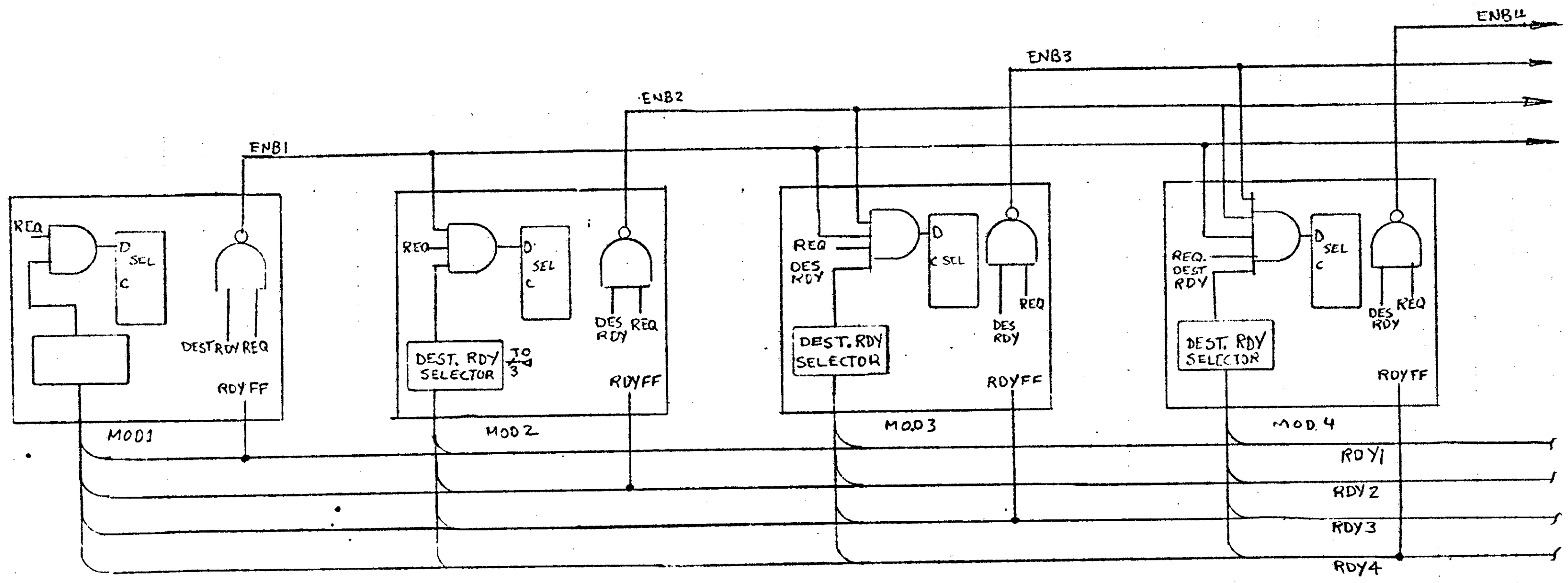
8.2 FUNCTIONAL DESCRIPTION

8.2.1 Transmit Operation

When a module decides to transmit a message to another module, it must present its MCU with either a LO REQUEST (LREQ) or a HIGH REQUEST (HREQ) and the module number of the destination module. It must then wait for the MCU to send it a SELECT (SEL) pulse. Module uses SEL as a go ahead signal to send the message to the destination. MCU depending on the modules specific requirement can be designed to handle LREQ and/or HREQ.

For LREQ, MCU checks two items:

(a) It checks the RDY line of destination module to see if it can accept a message or not (see figure 41). If the destination RDY line is true,



NOTE: The most time critical path in the MCU bus is when the following events line up

A- Module 1 wants to transmit to MOD3 on LREQ basis
B- Module 2 wants to transmit to MOD4 on LREQ basis
C- Module 3 and 4 go RDY on the same cycle

Module 1's ENB (ENB1) must go low soon enough to get to module 2 and disable it from getting selected. Such that two modules are not selected at the same time.

Figure 41. MCU Priority Network

the transmitting module must then pull its ENB line low to keep all lower priority modules from using the bus.

(b) It checks the ENB lines of all the higher priority modules to see if any higher priority module is waiting to transmit a message on the bus, if so, this module must stay away from the bus until all ENB lines of the higher priority modules are high, i.e., no higher priority module is waiting to use the bus (see figure 41).

If both (a) and (b) check positively, i.e., the destination module is ready and no higher priority module is trying to use the bus, this module will then receive the "SELECT" line from its MCU during the next clock cycle.

For HREQ, MCU checks item (b) only. This is because MCU expects the host module to use the HREQ when the destination module is "busy", e.g., during the CWA to memory, two messages must be transferred to the memory. The first message is the memory address and the second message is the data that must be stored in the memory at that address. After the address is transmitted to the memory, memory goes "busy" waiting for the data to arrive. The transmitting module must therefore use HREQ in order to be able to transmit the data to the memory.

Figure 42 shows typical interface lines between a module and its MCU.

8.2.2 Receive Operation

There are two receive modes:

(a) Expected receive mode, in which module is expecting to receive a message from another module. In this mode, the expecting module must store the module number of the module that it is expecting to receive a message from. When addressed, module must compare the stored module number with the content of FROM BUS (0,1,2) (containing the module number of the transmitting module). If they compare the message received is from the expected module (See figure 42).

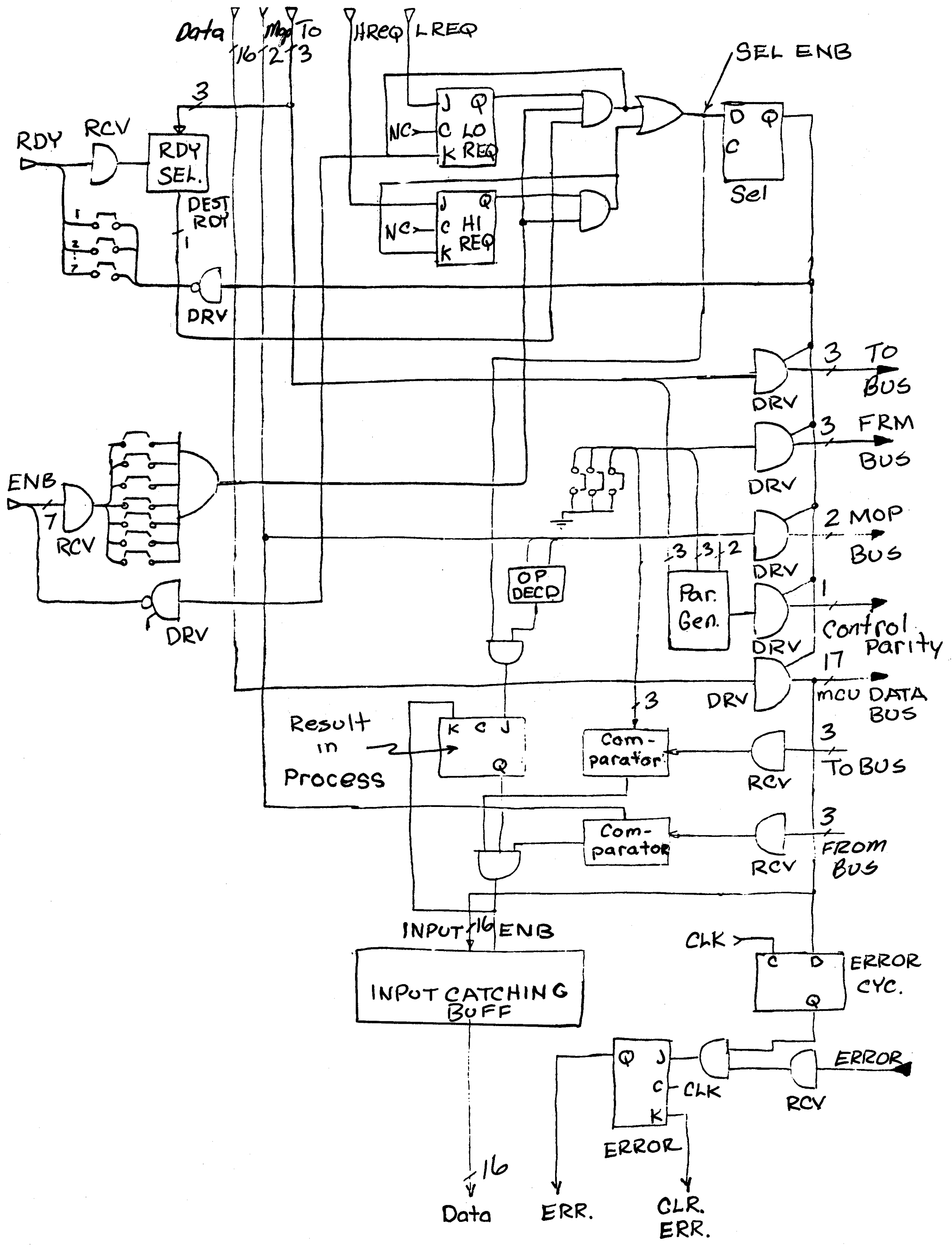


Figure 42. MCU Logic Block Diagram

(b) Unexpected receive mode. In this mode a module receives a message from another module when it is not expecting it. This mode, if applicable, causes different actions in different modules. For example, in memory modules, unexpected messages, if accompanied by non NOP opcode, is interpreted as the address for RWA or CWA.

8.3 MCU BUS DESCRIPTION

8.3.1 Ready

The RDY (1-7) lines indicate the busy/ready status of the modules numbers, one through seven. High state indicates the module is RDY. There is one RDY line per module. RDY lines are bi-directional. They are pulled low ("busy" state) by the transmitting MCU during the transmission cycle. This is done to keep the receiving module from getting selected on the cycle after it receives the message. It is, however, up to the receiving module to maintain its "busy" status thereafter, until the operation complete time, if it so desires. RDY lines can change state no later than 20 ns after the clock reference edge.

8.3.2 Enable

The ENB (1-7) lines are unidirectional. Each line is dedicated to a single module. Module uses its ENB line to disable the lower priority modules from using the bus by pulling it to the logical low state (see figure 41). The conditions that must be present before a module can pull on its ENB line is according to the following equation.

$$\text{ENB}(n) = \text{LREQ} \cdot \text{RDY (destination)} + \text{HREQ}$$
$$n = 1\text{-}7$$

The part of the MCU logic that pulls down on the ENB line must be designed such that the path after the RDY receivers to the input of the ENB driver is no longer than 20 ns.

8.3.3 TO Bus

The TO bus (0, 1, 2) carries the module number of the destination module (line 0 is the most significant bit). TO BUS is bi-directional, it is used by all modules. The module that is selected to use the MCU BUS in the current cycle must gate its destination onto TO BUS no later than 20 ns after the reference edge of the clock.

8.3.4 FROM Bus

This bus carries the module number of the source module (line 0 is the most significant bit). FROM BUS is bi-directional, and is used by all modules. The module that is selected to use the MCU BUS in the current cycle must gate its own module number onto TO BUS no later than 20 ns after the reference edge of the clock.

8.3.5 OPCODE Bus

The OPCODE bus (0, 1) is a bi-directional bus that carries the two bits opcode. Opcode can mean different operations to different modules, e.g., memory interprets these two bits as follows:

0, 0	=	NOP
0, 1	=	Write
1, 0	=	Read
1, 1	=	Read No Write

8.3.6 Control Parity

The CPAR line carries odd parity for combined TO, FROM, MOP buses. This parity is calculated by the transmitting module, and is checked by the receiving module. If an error is detected the receiving module must activate the Control Parity Error (CPE) line for one cycle, and ignore the current transmission. The transmitting module must gate CPAR no later than 20 ns after the reference edge of the clock.

8.3.7 MCU DATA Bus

The MCU DATA bus (0-16) is bi-directional and carries address, data, and control information between transmitting and receiving modules. This bus is 17 bits wide, of which the 17th bit is the odd parity on the first 16 bits. The parity must be generated on the 16 bits data by the transmitting module before the SELECT cycle, so that the 16 bits data and the one bit parity are gated onto the bus, during the select cycle, no later than 20 ns from the reference edge of the clock.

8.3.8 Address Parity Error

This line is bi-directional and is asserted by the receiving module, for one cycle, during the cycle after the message is received, provided: (a) message had parity error and (b) module is designated to recognize the error and flag the error back to the transmitting module. For example, memory module interpret the first cycle that it is addressed as the address cycle. During this cycle memory looks for an address parity error and flags it back to the transmitting module, and internally it ignores the message.

8.3.9 CLOCK

This line carries the system clock. $\overline{\text{CLOCK}}$ is a symmetrical square wave. The falling edge of the $\overline{\text{CLOCK}}$ is defined to be the reference edge. System modules must use this signal in all their synchronizing logic and their MCU logic.

8.3.10 Master Reset

This line is normally in the high state. It goes low when either front panel system reset button is depressed, or when any of the system DC voltages reach below the specified threshold voltages. After power first goes on MASTER RESET remains low for approximately 500 μsec.

Modules must use MASTER RESET to reset all their state FF's.

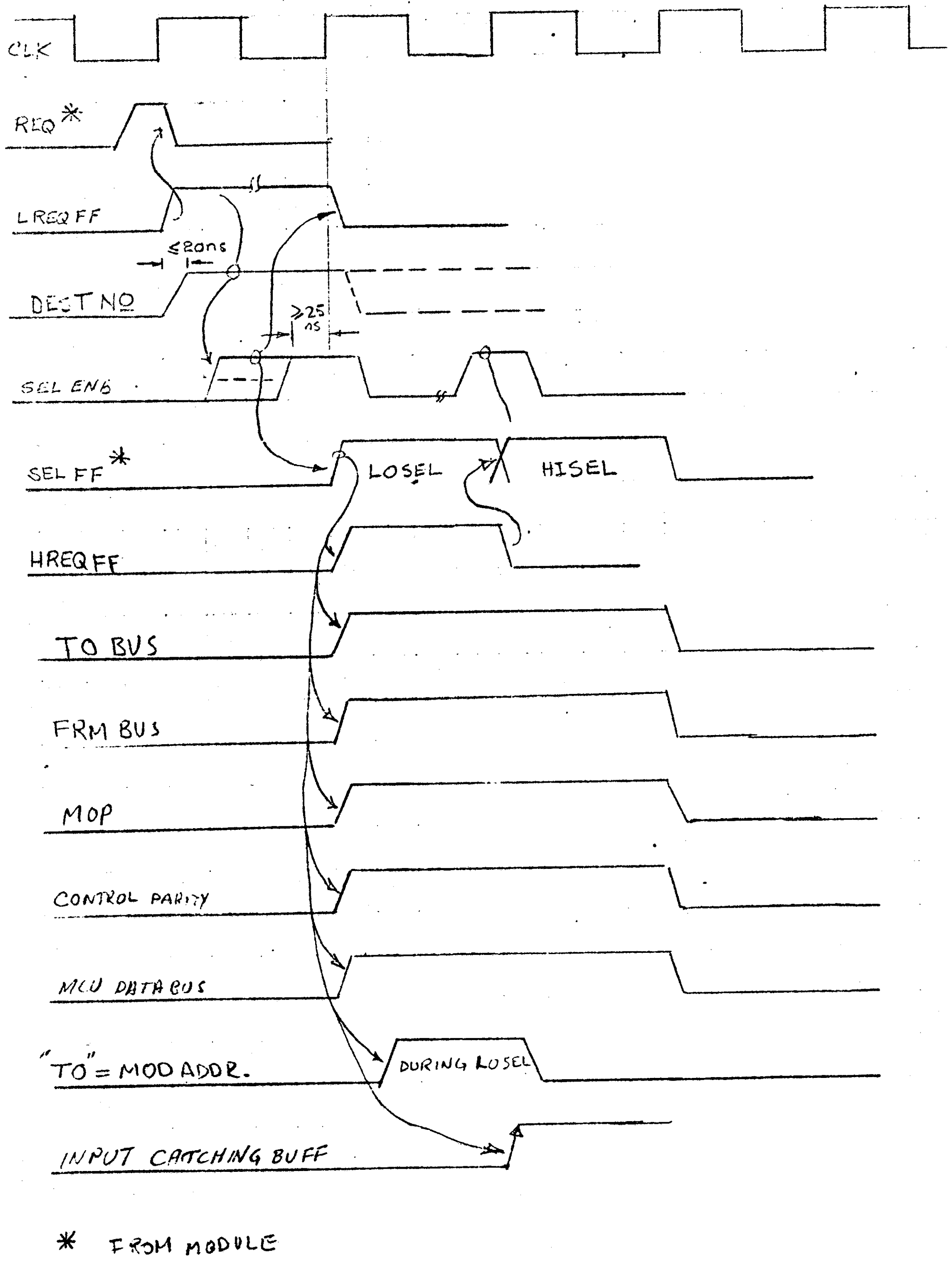


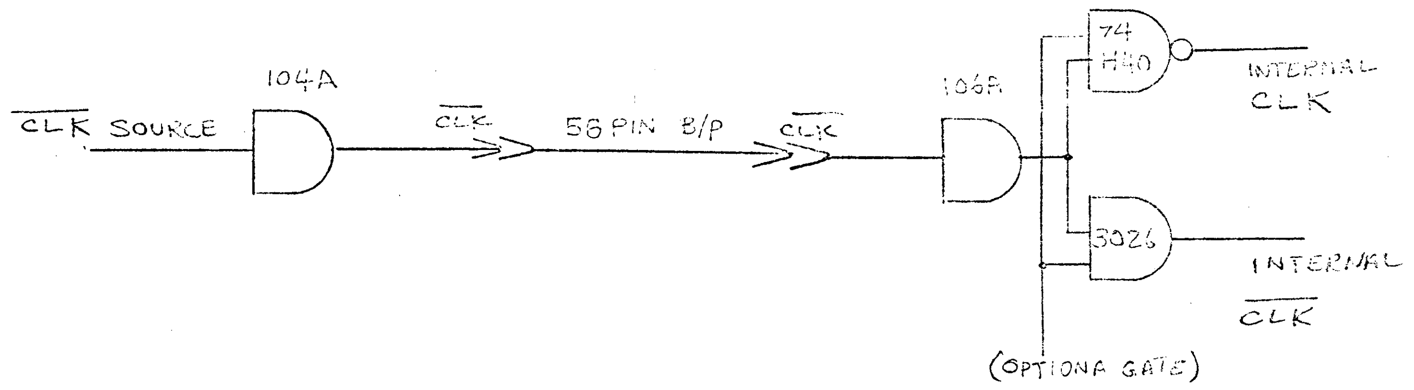
Figure 43. MCU Timing Diagram

8.4 ELECTRICAL AND PHYSICAL DESCRIPTION

MCU line are driven by HP 104A TTL Drivers and are received by HP 106A TTL Receivers.

MCU cable is a 50 wire ribbon cable that is connected to the P2 connector. The characteristic impedance of each line is approximately 25 ohms. The cable is terminated at both extreme ends. The terminating network on each end consists of two 1K resistors, one connected to +5V and one connected to GND.

Clock signal is available to all system modules via the 56 pin power connector. This signal should be received by each module with no more than one HP 106A receiver. This is to avoid any excessive AC loading of the clock signal. For clock to match in different modules, the clock receiver in each module must be cascaded with inverting and non-inverting gates to obtain CLOCK and $\overline{\text{CLOCK}}$ respectively.



MCU Clock

SECTION 9

HIGH SPEED SELECTOR CHANNEL

9.1 INTRODUCTION

The Alpha High Speed Channel (HSC) is designed to complement the SIO I/O system of the Alpha. It provides compatibility for high speed I/O devices to interface directly with the Alpha MCU Bus resulting in higher transfer rates than provided by the SIO system. Hardware and software compatibility with the SIO system is maintained, providing versatility and interchangeability. The HSC executes all SIO order pairs, provides buffering and control of the device controller and interfaces with memory.

Each Selector Channel comprises two logic boards, one containing the control logic and the other containing the registers, counters and buffers required for the execution of the SIO program. The two boards communicate using the three connectors across the top of the boards. Each Selector Channel is capable of interfacing with up to 8 device controllers in a shared manner. That is, only one I/O program can execute at any one time. All other devices must wait until the program executing either Ends or Clear Interface is initiated by the device. At this time, one of the other devices may initiate an I/O program. The device executing has complete and dedicated control of the HSC until the program completes or is terminated by some other means, e.g. clear interface.

The HSC communicates with the device controller via the Channel Cable. This 50 connector ribbon cable contains a data path to and from the device (not the same as the IOP Bus) and control signals to and from the device. The cable connects to all those devices which are interfaced to the HSC. See Section 9.9

The HSC interfaces with the MCU Bus through a logic card called the Port Controller. The Port Controller multiplexes 4 individual HSC onto the MCU Bus. The multiplexing is accomplished by assigning two priority levels within the Port Controller which process HSC requests for MCU Bus service.

Each HSC has a potential transfer rate of 2.0 M words/sec maximum. The actual rate will be determined by the device and memory speeds. Device controllers that interface with the HSC require the Channel-SIO Bus Logic option. This provides for compatibility with the I/O Bus and the Multiplex SIO Board.

9.2 LOGIC FUNCTIONS

A convenient method to understand the operation of the HSC is to break the overall HSC into a group of smaller logic functions or controllers. These controllers perform some function, e.g. fetch next order pair, each time they are initiated. Then by understanding each function, the overall/channel operation is easily formulated and understood.

The logic functions are:

- Initiator Section
- Fetch Sequencer
- Device Controller Interface
- Pre-fetching Sequencer
- Program Execution Controller
- Terminator Section
- Error Controller

Often during the explanation of the HSC two phrases will appear; first, Channel SO cycle, and second, Transfer Cycle. These terms are defined in Section 9.6.

9.2.1 Initiator Section

Each device controller requesting to start an I/O program must pass through the Initiator Controller once. When the device controller decodes the SIO direct command and initiates a request to the HSC, the HSC sets its Active FF and makes $\overline{\text{Enable}}$ false. No other device controller can now initiate its I/O program as long as the Active FF is set. When the device controller initiated the request, it also puts its device number multiplied by 4 and its select number onto the channel Bus.

DEV # * 4 is the address of the DRT where the I/O Program Counter (IOPCNT) resides, and the select number is the effective address of the device running its I/O program.

When the Active FF sets, the Initiator sequencer generates DEV # and Select Enables to load their respective registers, enables the DEV # Register onto the Channel Address lines and initiates a Transfer Cycle Outbound. The sequencer then returns $\overline{SI}$ to the IOP for the device controller and waits for Request to be removed by the IOP. When Request is removed the Initiator sequencer enables the Fetch Sequencer. The Initiator Section is now complete.

When the Transfer Cycle Outbound was initiated the DEV # * 4 was transferred to memory and the IOPCNT was returned as data and loaded into IOPCNT Counter Register. The Transfer Cycle completes when the IOPCNT is loaded.

The Select number resides permanently in the Select Register. It is encoded and becomes active lines on the Channel Bus to the device. Valid select numbers are 0-5, 14, 15.

9.2.2 Fetch Sequencer

The Fetch Sequencer is used to fetch order parts from memory except for prefetching. Whenever the orders Read or Write are executing and chaining is true, the Fetch Sequencer is disabled. But in all other cases, this controller is used for getting the next order to be executed. The sequencer is conditioned to start when the following condition is met. The presently executing order is completed "And" no Transfer Cycle is in progress "And" the initiator section is completed "And" the last order executed was not Read, Write or Command, "And" the presently executing order is not Read or Write with Chaining. If the last order executed was Read without chaining, Write without chaining, or Command, the Fetch Sequencer will start only after the above conditions are met and a Device Service Request is generated by the device controller.

When the conditions are met, the Fetch Sequencer initiates a Transfer Cycle Outbound and enables the IOPCNT onto the Channel Address lines. This transfer Cycle is for the I/O Command Word (IOCW). When data is returned from memory it is loaded into the IOCW Register Active, and the IOPCNT is incremented. The Fetch Sequencer then initiates a second Transfer Cycle Outbound and enables the IOPCNT onto the Channel Address lines. This Transfer Cycle is for the I/O Address Word or Operand (IOAW). When data is returned from memory, it is loaded into the IOAW Register Active, and the IOPCNT is incremented. A flag is set (the order complete FF is reset) at this time indicating to the Program Execution Controller that a new order is present and execution can begin.

9.2.3 Device Controller Interface

This logic section contains the signal lines that interface with the device controller. The lines are defined below and timing diagrams appear in Appendix II. These diagrams are also applicable for the Program Execution Controller. Also see Section 9.9.

$\overline{\text{Device Service Request}}$

This signal is used by the device controller to initiate a CHAN SO cycle for data transfer or to indicate that the next order pair should be fetched following the execution of Read, Write or Command Order.

$\overline{\text{ACK SR}}$

Unlike the MUXSIO Logic, the HSC does not generate $\overline{\text{ACKSR}}$. This signal used by the device controller to reset the device SR FF is generated in the bus logic. The device can use this signal (Chan Ack) to clear his own SR FF or inhibit it.

$\overline{\text{CHAN SO}}$

Signal used to initiate the Channel SO cycle. It is used by the device controller in conjunction with the select lines to enable the input buffers to the channel bus. CHAN SO remains true until $\overline{\text{CHAN ACK}}$ is returned by the device controller.

$\overline{\text{CHAN ACK}}$

This signal is returned by the device controller in response to the following signals, $\overline{\text{Read Strobe}}$, $\overline{\text{Write Strobe}}$, $\overline{\text{CMD Strobe}}$, $\overline{\text{PCMDI Strobe}}$, $\overline{\text{Sense Strobe}}$, In and Out XFER Toggles, $\overline{\text{Set Int}}$, $\overline{\text{Set Jump}}$, $\overline{\text{Read Next Word}}$, $\overline{\text{WCDV}}$.

The device controller may use $\overline{\text{CHAN ACK}}$ to reset the device SR FF.

CLOCK

The HSC provides a version of the system clock to be used by the device controller to clock the SERVICE REQUEST FF. The SR FF should be a negative edge triggered device. See Section 9.11 for a suitable service request circuit.

$\overline{\text{DEV END}}$

During the execution of a Read or Write order, the device may prematurely terminate the order by asserting $\overline{\text{DEV END}}$. $\overline{\text{DEV END}}$ will cause the following condition to occur for Read and Write orders.

DEV END and Write

A. If Chaining:

1. Write strobe is not generated
2. OUT XFER will not toggle
3. DEV END FF will terminate Channel So cycle by generating $\overline{\text{WCDV}}$.

*4. Wait for DEV SR for next transfer.

*When chaining, the pre-fetch controller is on and the next write order is pre-fetched and waiting in buffers until the actively executing order is complete.

B. If not chaining:

1. DEV END will terminate Channel SO cycle.

2. Write Strobe is not generated.

3. OUT XFER Toggle is generated to clear the device OUT XFER FF.

DEV END and Read

A. If chaining:

1. In XFER will not toggle

2. Read next word will be generated for the $\overline{\text{DEV SR}}$ pending.

3. DEV END FF will terminate the Chan SO cycle from Read Next Word.

*4. Wait for DEV SR for next transfer

B. If not chaining:

1. DEV END FF sets, aborting Read next Word.

2. DEV END will terminate the CHAN SO cycle.

3. In XFER toggle is generated to clear the device IN XFER FF.

The $\overline{\text{DEV END}}$ input to the channel is only active during the execution of Read or Write orders.

IN XFER TOGGLE

Signal generated by HSC to toggle the IN XFER FF in the device controller. The device IN XFER FF is clocked on the trailing edge of $\overline{\text{CHAN SO}}$.

OUT XFER TOGGLE

Signal generated by HSC to toggle the OUT XFER FF in the device controller. The device OUT XFER FF is clocked on the trailing edge of $\overline{\text{CHAN SO}}$.

$\overline{\text{REQ}}$

This line is asserted by the device controller when its I/O program is to be executed by the HSC. The device controller asserts $\overline{\text{REQ}}$ in response to the direct I/O command SIO.

$\overline{\text{CLR IL}}$

Clear Interface is a line used by the device controller to prematurely terminate its I/O program. The HSC accepts $\overline{\text{CLR IL}}$ only following the active FF being set, and then immediately terminates the I/O program. The HSC also will restore the current value of the IOPCNT into the DRT at DEV # * 4 in responses to $\overline{\text{CLR IL}}$.

The signals $\overline{\text{REG}}$ and $\overline{\text{CLR IL}}$ are time-shared on the same line from the device controller. The state of the Active FF in the HSC is used to differentiate between $\overline{\text{REQ}}$ and $\overline{\text{CLR IL}}$.

$\overline{\text{ENABLE}}$

Enable is similar to the SIO multiplexed signal $\overline{\text{SIO OK}}$. When no I/O program is executing in the HSC, $\overline{\text{ENABLE}}$ is asserted. As soon as $\overline{\text{REQ}}$ is asserted and the channel goes active, $\overline{\text{ENABLE}}$ is made false.

Upon executing the order END or upon receipt of $\overline{\text{CLR IL}}$, $\overline{\text{ENABLE}}$ is again asserted.

$\overline{\text{SET JMP}}$

Signal generated by the channel during the execution of a conditional jump order to clock the device Jump FF. The device conditions this FF and returns the $\overline{\text{JMP MET}}$ line to the channel.

$\overline{\text{JMP CONDX MET}}$

The HSC interrogates this input following generating the $\overline{\text{SET JMP}}$ clock. If this line is true then the HSC transfers the IOAW to the IOPCNT and initiates a Transfer Cycle

Outbound for the next IOCW. If $\overline{\text{JMP MET}}$ is false, then a Transfer Cycle is initiated without updating the IOPCNT from the IOAW.

$\overline{\text{SET INT}}$

This strobe line is used to set the device interrupt FF when the Int. Order executes. If the MSB of WC Register is one when executing an END order, the $\overline{\text{SET INT}}$ strobe is generated.

$\overline{\text{SENSE STROBE}}$

This strobe line is asserted during the execution of the Sense Order and the End Order. The device controller returns the contents of its status register in response to the $\overline{\text{SENSE STROBE}}$ and generates $\overline{\text{CHAN ACK}}$.

$\overline{\text{RD STROBE}}$

The strobe line is asserted during the execution of the Read Order and the following a device generated service request. The device controller returns the contents of its data in buffer and generates $\overline{\text{CHAN ACK}}$.

$\overline{\text{READ NEXT WORD}}$

A signal used in conjunction with the $\overline{\text{RD STROBE}}$. It is used to indicate to the device controller that a Read Order is beginning and to inform the device controller that it can read a new word from the device.

$\overline{\text{WRITE STROBE}}$

This line is asserted by the channel during the execution of the Write order and in response to a device generated service request. The HSC puts data onto the channel bus and when $\overline{\text{WRITE STROBE}}$ is asserted, the device controller can accept the data. The device controller returns CHAN ACK which terminates the $\overline{\text{WRITE STROBE}}$ and then the data. Data is not valid at the leading edge of this strobe.

$\overline{\text{PCMD1}}$

The Programmed Command 1 strobe indicates to the device controller that the first word of Control information is on the Channel Bus. The device controller must return $\overline{\text{CHAN ACK}}$ and generate a device service request for the second word of control information. Data is not valid at the leading edge of this strobe.

$\overline{\text{CMD STROBE}}$

This strobe indicates that the second word of control information is available to the device controller. It is asserted following a device generated service request. The device controller must return $\overline{\text{CHAN ACK}}$ and also generate device service request for the program to continue execution. Data is not valid at the leading edge of this strobe.

$\overline{\text{XFER ERROR}}$

The $\overline{\text{XFER ERROR}}$ line from the HSC to the device controller indicates that an error has occurred. The $\overline{\text{XFER ERROR}}$ is generated by the following conditions:

a. Illegal Address – The Address Lines contain an address larger than that which is permitted.

b. Address Error – The memory has detected a Parity Error on the last address sent to memory from the HSC.

c. System Parity Error – the HSC Port Controller has detected a control parity error.

d. Data Parity Error – The HSC has detected a parity error on the data transferred from memory to the channel.

Any of the above conditions will cause the HSC to initiate a Chan So cycle to transfer $\overline{\text{XFER ERROR}}$ to the device controller. The device controller will then generate a $\overline{\text{CLR}}$ IL signal terminating its I/O program and generate an Interrupt. The HSC will terminate the program, restore the IOPCNT to memory and re-assert $\overline{\text{ENABLE}}$

permitting another device controller to initiate the channel with a REQ for service to start an I/O program.

See Section 9.9 for the Channel Bus interchange drawing. This drawing gives all the signals and direction that are used for communication between the HSC and its respective device controllers.

$\overline{\text{EOT}}$

The signal $\overline{\text{EOT}}$ (End of Transmission) is passed to the device controller during the CHAN SO cycle for the last data word. The word count register is not incremented for DEV END.

9.2.4 Pre-fetching Sequences

The Pre-fetching sequencer is activated whenever Read or Write orders with chaining are executing. To enhance the speed, and to make the transition between orders when chaining is taking place smoother, the next order to execute is fetched and stored in buffer registers until the presently executing order is complete. The prefetching takes place during the data transfers conditioned on two simple rules.

Data transfers during Read and Write orders utilize two buffers. For Write Orders, the HSC is actively keeping both buffers full independently of the device controller. For Read orders, the HSC is attempting to keep to the input buffers empty by transferring their contents to memory.

If chaining is active and a Write is executing, a Pre-fetch cycle will initiate only when both output buffers are full. If a Read order is executing, a Pre-fetch cycle will initiate only when both input buffers are empty.

The Pre-fetch Sequencer must initiate two Transfer Cycles Outbound, the first for the IOCW and the second for the IOAW. The stated conditions pertaining to the input and output buffers in the channel must be valid for both Transfer cycles.

The Pre-fetch sequencer generates the enables to load the buffer registers where the IOCW and IOAW are temporarily stored. When the executing order completes and both data buffers are empty, the logic generates a set of enables to transfer the IOCW and IOAW buffers into the active registers and sets the flag indicating that execution can continue.

For Write orders, when the order completes, the OUTPUT data buffer flags are set to empty so that the pre-fetch sequencer can transfer the buffers to active, but when the Read Order completes, valid data will be present in the input data buffers. One or two Transfer Cycles Inbound will be generated to unload the input data buffers until both input flags are reset. Only when the input flags are reset will the set of enables to transfer the IOCW and IOAW buffers into the active registers be generated.

The inputs into the Pre-fetch Sequencer indicate the state of the order buffers, empty or full, and also indicate the state of the data buffers, empty or full. These signals steer the logic to permit the fetching of the order pair, interleaved with the transferring of data.

9.2.5 Program Execution Controller

The Program Execution Controller encompasses the logic which decodes the order, interfaces with the Device Controller Interface, and initiates Transfer Cycles for data transfer. Also, when the order completes, the Program Execution Controller conditions the fetch sequencer to start for the next order pair.

The Program Execution Controller (PEC) initiates whenever the order complete flag is reset indicating that the next order has been fetched and is in the active buffers ready for decoding and execution.

The PEC decodes and executes eight orders: Read, Write, Control, Jump, END, Sense Interrupt and Return Residue. Timing diagrams for these eight orders are found in Section 9.7.

Following completion of the order or $\overline{\text{DEV END}}$ in the case of Read or Write, the order complete FF sets indicating that the PEC has completed.

The PEC also initiates Transfer Cycles for data transfer. For example, following the completion of the Sense Order the status information read from the device controller must be transferred to memory.

For Read, Write and Control, CHAN SO Cycles are initiated by the device control. For Int, Jump, Sense, and END the CHAN SO cycles are initiated by the HSC. Return Residue requires no CHAN SO cycle for execution.

9.2.6 Terminator Section

The Terminator Section responds to two inputs, the END order and the device controller generated CLEAR Interface, $\overline{\text{CLR IL}}$. The Terminator Section must transfer the IOPCNT to the DRT and re-enable device controllers to initiate new SIO programs.

The END order resets the Order Complete FF and transfers status to memory. The $\overline{\text{CLR IL}}$ signal from the device controller is asynchronous to the sequencing of the HSC. To make both inputs to the Terminator Sequence similar, the $\overline{\text{CLR IL}}$ FF in the HSC generates the condition to reset the Order Complete FF.

The following sequence of events occur in the Terminator Section when initiated.

Fetch Sequencer Inhibited from starting.

Order Complete FF resets ($\overline{\text{CLR IL}}$ only. For END, the Order Complete FF has already been set).

Channel Active FF and $\overline{\text{CLR IL}}$ FF are reset.

When the Channel Active FF resets, the Fetch Sequencer is hard reset.

The Initiator Sequencer steps to its reset state removing the qualifiers on the Fetch Sequencer.

As soon as the TIP (Transfer In Progress) FF resets, then a Transfer Cycle is initiated to restore the IOPCNT in the DRT; A general reset is made.

9.2.7 Error Controller

The Error Controller has four inputs which indicate error.

ILLEGAL Address

Address Parity Error

System Parity Error

DATA Parity Error

These errors require the device controller to generate an Interrupt and $\overline{\text{CLR IL}}$ terminating its I/O program. A CHAN SO cycle is required to transfer the $\overline{\text{XFER ERROR}}$ signal to the device controller.

When the Error Controller detects one of the errors listed above, it conditions the $\overline{\text{XFER ERROR}}$ true and initiates a CHAN SO cycle. It is possible that the HSC may be presently in a CHAN SO cycle. To insure that the device controller XFER ERROR FF is set, the Error Controller permits both CHAN SO cycles to execute concurrently, therefore, only one clock is generated ensuring that the device controller recognizes the error conditions. The concurrent cycle is important. Informing the device controller as soon as possible that an error has occurred permits the device controller to take whatever action is required to protect the integrity of the system.

9.3 GENERAL SPECIFICATIONS

2.0 MEGAWORD/SEC maximum transfer rate; Typical Rates, 2.0 MHZ Read
1.7 MHz Write

SIO Hardware and Software compatibility

Executes READ orders
WRITE
CONTROL
SENSE
INTERRUPT
JUMP

RETURN RESIDUE

END with optional interrupt

Requires the Channel-SIO version of the Bus Logic.

Fully double buffers for input and output

Generates $\overline{\text{XFER ERROR}}$ for the following conditions:

Address Parity Error

System Parity Error

Data Parity Error

Illegal Address

Up to 4 HSC can be multiplex through one port onto the MCU Bus.

IOPCNT Resident in the HSC – Restored for END orders and device generated $\overline{\text{CLR IL}}$.

Operates in a shared mode, with a maximum of 8 device controllers sharing one High Speed Selector Channel.

9.4 PORT CONTROLLER

The Port Controller provides for multiplexing four HSC onto MCU Bus through a single port or module number. Each HSC requests to the Port Controller for service and indicates the nature of the request, Read Write or Clear Write. The Port Controller resolves priorities, both the Alpha MCU Bus priorities and the internal priorities between the individual HSC.

The Port Controller generates Control Parity on the control lines, MOP, TO, FRM and checks parity on the same set of lines. It also monitors the Address Parity Error from memory and responses by transferring the state of the line to all the HSC Error Controllers.

The Port Controller MCU Bus interface is similar to that used in the IOP.

See Section 9.10 for the Port Controller Bus Interchange.

Two priority levels are established in the HSC to facilitate the throughput in the Port Controller. All Low Requests made to the Port Controller are arranged in priority from 1 to 4, with 1 being the highest, and 4 the lowest. All Low Requests are processed in order according to this scheme.

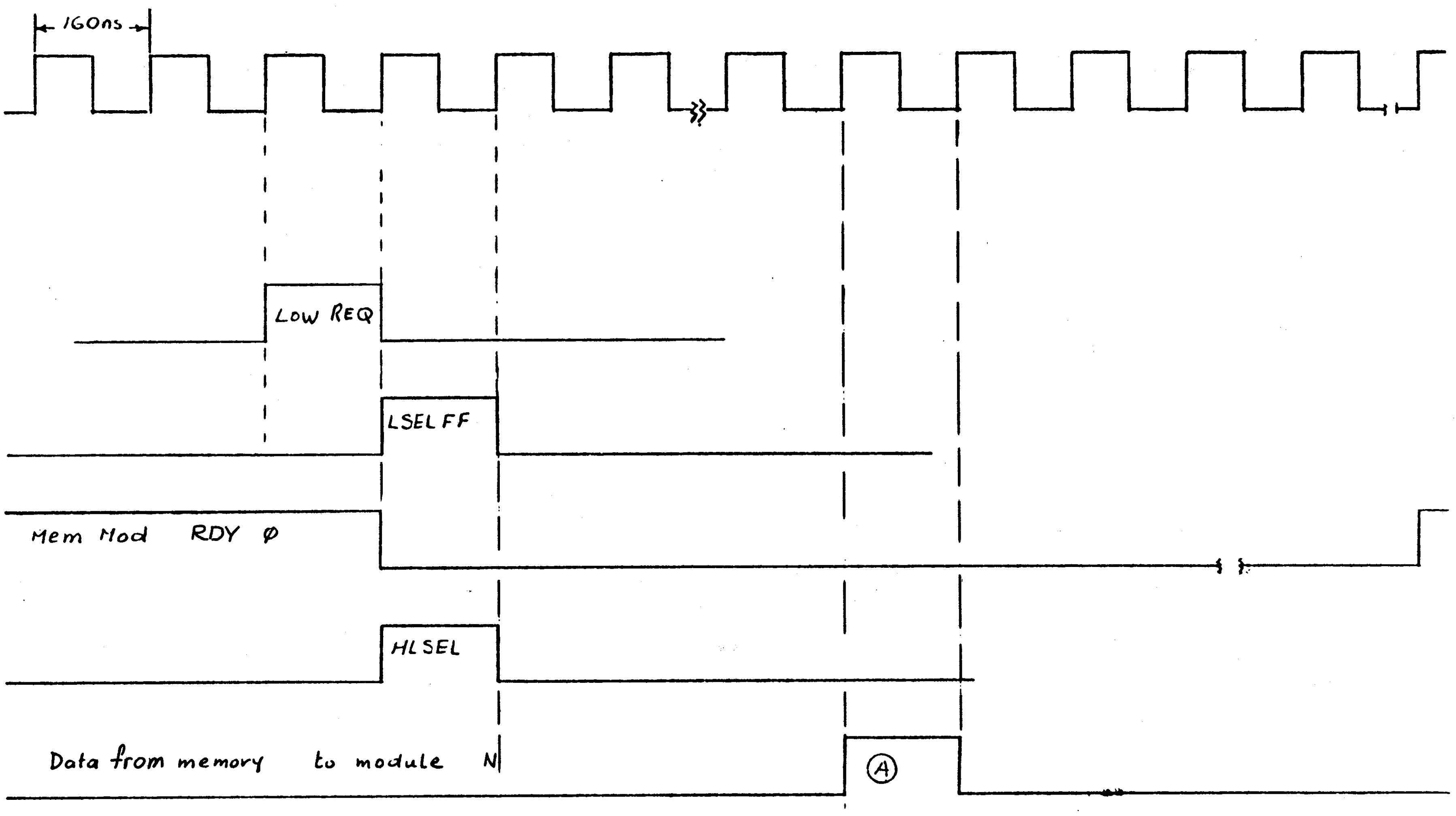
If one of the Low Requests is accompanied by a High Request (CLEAR Write), as soon as the Low Request is processed, the High Request will override all pending Low Requests. The High Request will execute when enabled. The next Low Request with highest priority will be enabled as soon as the High Request completes. A typical MCU bus transfer is shown in figure 44.

9.5 HIGH SPEED SELECTOR CHANNEL DIAGNOSTIC

The HSC does not interface with the Alpha CPU directly. It provides a path from the device controller to and from memory, but does not interface with the IOP. Diagnostics could be run indirectly if a device controller or test fixture provided the link between the HSC and the IOP Bus. The test fixture could be programmed in the direct Read/Write mode to provide the various inputs and outputs for the HSC. This would permit the HSC to run test diagnostics.

To maximize the test fixture's usefulness, it must be able to interface with all the signals in the Chan. Bus and provide conditions like CLR IL, DEV END, SERVICE REQUEST, check the STATE of XFER FF, respond to the programmed strobes and the XFER ERROR condition, and provide a jump FF which can be conditioned true and false.

It must also have a data register which can be loaded and read with both the SIO programmed strobes and the direct Read/Write Strobes, control logic to generate device service requests and the Channel-SIO version of the bus logic.



Ⓐ Data from memory is transfered into HSC, because HSC recognizes that memory module Ø was addressed, and because it is module #N. Sys Parity is Checked during this cycle.

Figure 44. Typical MCU Bus Transfer via Port Controller

9.6 COMMUNICATION AND TRANSFER CYCLES

9.6.1 Channel SO Cycles

Throughout the execution of the I/O program, the HSC must communicate with the device controller over the Channel Bus. The communication may take many forms, transfer data to or from the device controller, set the Interrupt FF in the device controller or interrogate the status of the Jmp FF in the device controller. Also, the signal XFER ERROR must be passed onto the device controller when the HSC detects that the condition has occurred. Whenever the HSC communicates with the device controller, a Channel SO cycle will take place.

This cycle can be initiated either by the device by setting the Service Request FF or by the HSC. When the cycle is initiated, CHAN SO is asserted and all pending strobes or control signals are released to the device controller. When the device controller receives $\overline{\text{CHAN SO}}$ and providing it is not being addressed by the IOP, the Channel Bus Logic will return $\overline{\text{CHAN ACK}}$. $\overline{\text{CHAN ACK}}$ is generated from all the strobes and control signals provided by the HSC. $\overline{\text{CHAN ACK}}$ indicates to the HSC that the cycle may terminate. All signals (except data) sent to the device controllers return $\overline{\text{CHAN ACK}}$.

For data transfers (Read or Write), the device Service Request FF is used in addition to requesting service to win priority usage of the bus logic over the IOP.

9.6.2 Transfer Cycle

Each HSC communicates with the Port Controller through the Transfer Cycle. This cycle initiates the Port Controller into starting a memory transfer from the HSC. The HSC generates Transfer Cycles whenever data is to be transferred to or from memory, to fetch order pairs, to fetch the IOPCNT from the DRT, and to restore the IOPCNT in the DRT.

The Transfer Cycle can be one of two forms; Inbound or Outbound, see table.

MOP	0	1	
CLEAR WRITE	0	1	INBOUND
READ WRITE	1	0	OUTBOUND

MOP is defined as Memory Operation, two lines from the HSC to memory indicating what function memory is to perform. The HSC will never initiate a Transfer Cycle if one is presently in progress. The cycle terminates whenever data is received from memory (STROBE) or whenever data is transferred to memory (HSEL). The HSC terminates its Transfer Cycle before memory becomes ready in single memory systems. This permits maximum memory usage by the HSC by initiating a new Transfer Cycle even before memory becomes ready.

9.7 CYCLE TIMING

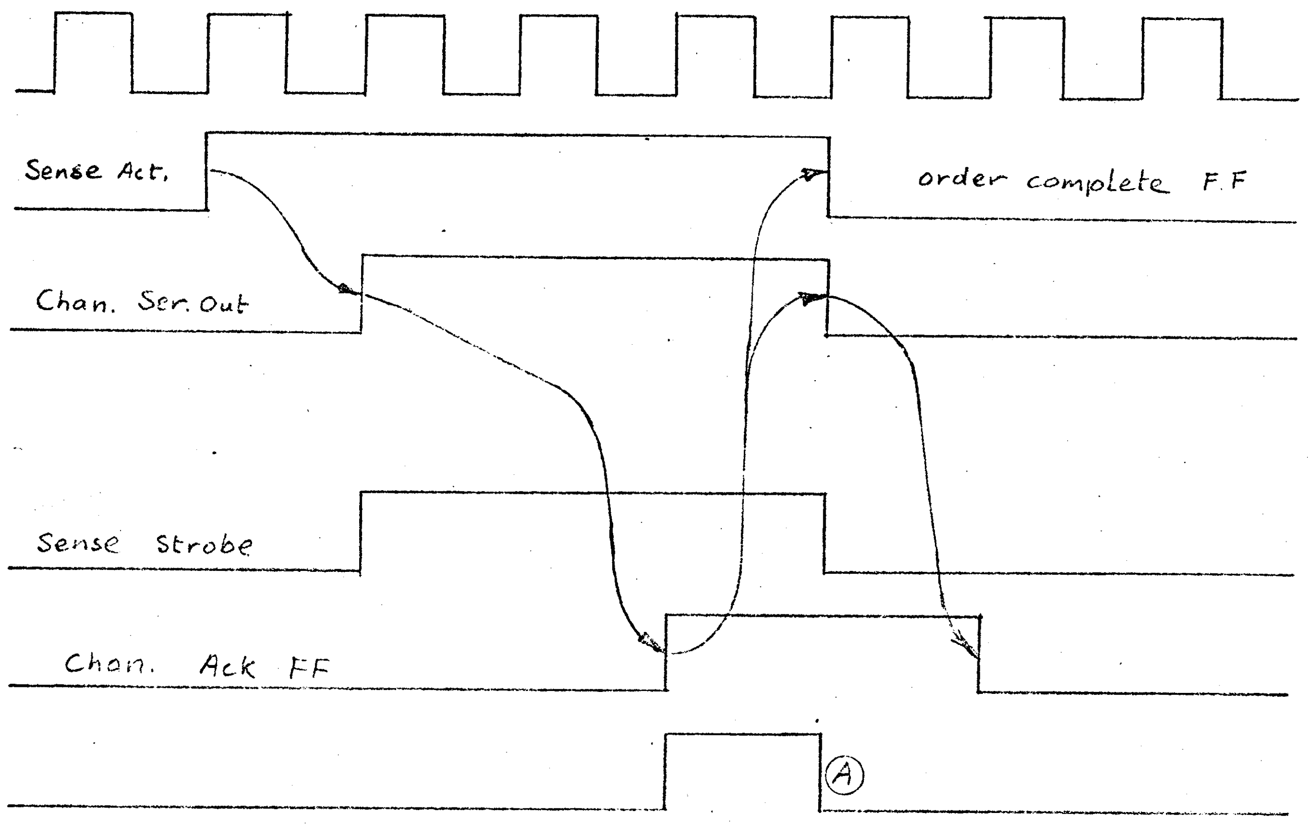
The timing diagrams for the eight orders executed by the HSC are shown in figure 45. Also points are indicated to show where Transfer Cycles begin. The timing diagrams show the occurrence of the CHAN SO cycle for those orders requiring communication with the device controller. The clock is the basic Alpha System Clock, 160 ns period.

9.7.1 Sense

The data from the Sense order is loaded into the memory location following the IOCW. To accomplish this, the IOPCNT is counted down by one and enabled onto the Channel Address lines. The Transfer Cycle initiated is Inbound with the IOPCNT as the address.

9.7.2 Return Residue

This order requires no device controller interaction. Whenever a new order is transferred into the active register, the WC (word count) is STORED for the last order executed. This permits the Return Residue order to read the WC register for the last order. See figure 46.



(A) 1. Data is Loaded into data in buffer
2. Transfer cycle Initiated.
3. IOPCNT is counted dwn by one.

Figure 45. Order Timing (HSC)

Returns Residue like Sense returns the data (SC Register) to the location following the IOCW. When the Transfer Cycle is initiated, the IOPCNT is counted down by one and enabled onto the Channel Address lines.

9.7.3 Interrupt

The Interrupt order does not return $\overline{\text{CHAN ACK}}$. The HSC must generated this condition internally to complete the CHAN SO cycle. No data is transferred for the Interrupt Order.

The Transfer Cycle enables the IOPCNT to the Address lines for the fetching of the next IOCW.

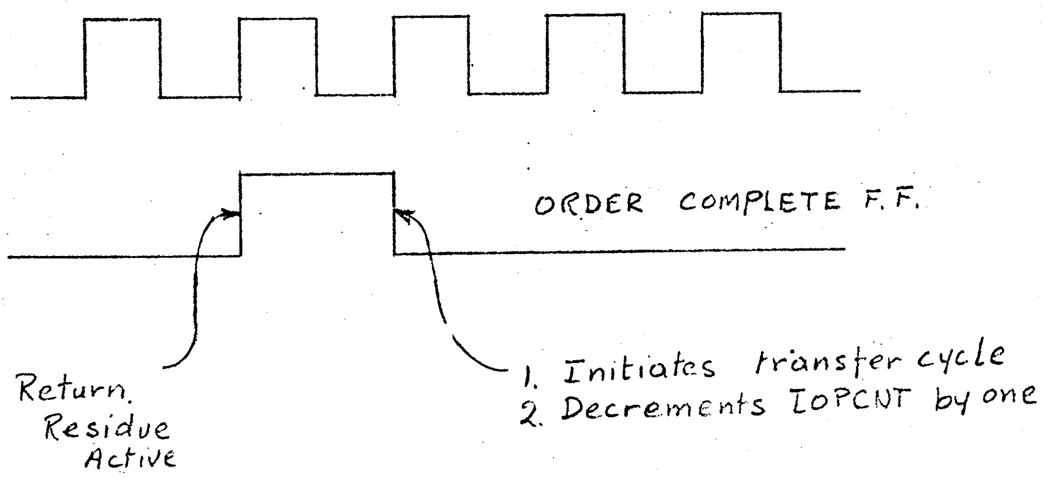


Figure 46. Return Residue Timing (HSC)

9.7.4 END

The End Order executes identically as the Sense order. In addition, if the MSB of the WC register is set to one, then the END order generates an Interrupt Strobe, END returns status to the location following the IOCW. END also conditions the Terminator section to start which transfers the IOPCNT to the DRT and puts the channel in the inactive state. At completion of the End order and with the data transferred to memory, the IOPCNT contains the address of the location END+2.

9.7.5 JUMP

Both unconditional and conditional jumps are permitted. Conditional jumps test the status of the device controller Jump FF for a valid condition met. The Set Jump Clock used to interrogate device controller Jump FF does not return $\overline{\text{CHAN ACK}}$. The HSC must internally generate the $\overline{\text{CHAN ACK}}$.

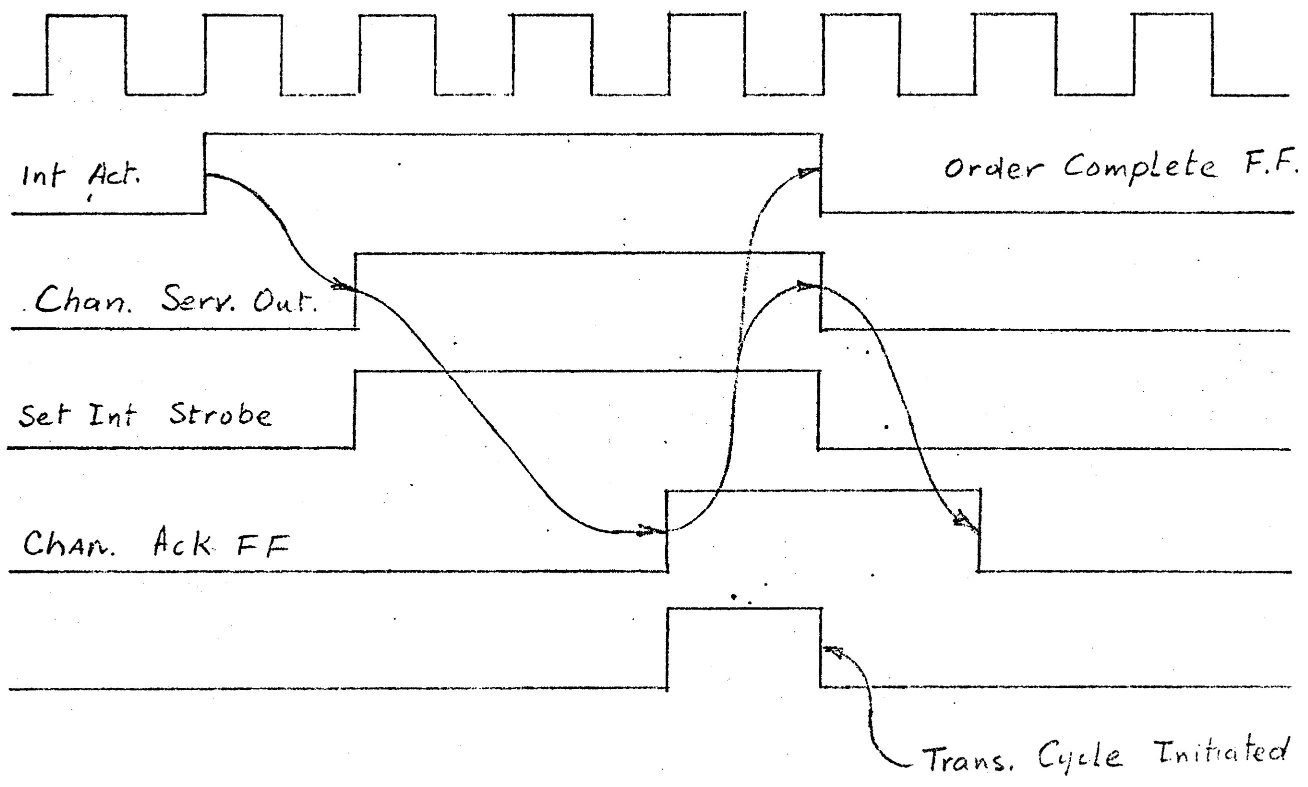


Figure 47. Interrupt Timing (HSC)

Figure 48 shows a conditional jump. Here, the Jump Met line is true. If False, then B1 would not be generated, i.e., the IOPCNT would not be updated.

A: The Jmp Met line from the device controller is sampled at time A. If it is true the Jump Met FF sets which transfers IOAW → IOPCNT and initiates the Fetch Sequencer.

Figure 49 shows an unconditional jump. This jump does not require a CHAN SO cycle—The IOAW is transferred to the IOPCNT and the Fetch Sequencer is initiated.

9.7.6 Command

The command order issues two strobes, $\overline{\text{PCMDI}}$ and $\overline{\text{CMD STROBE}}$. The first strobes the contents of the IOCW onto the data lines to the device controller. The second strobes the contents of the IOAW word onto the data lines to the device

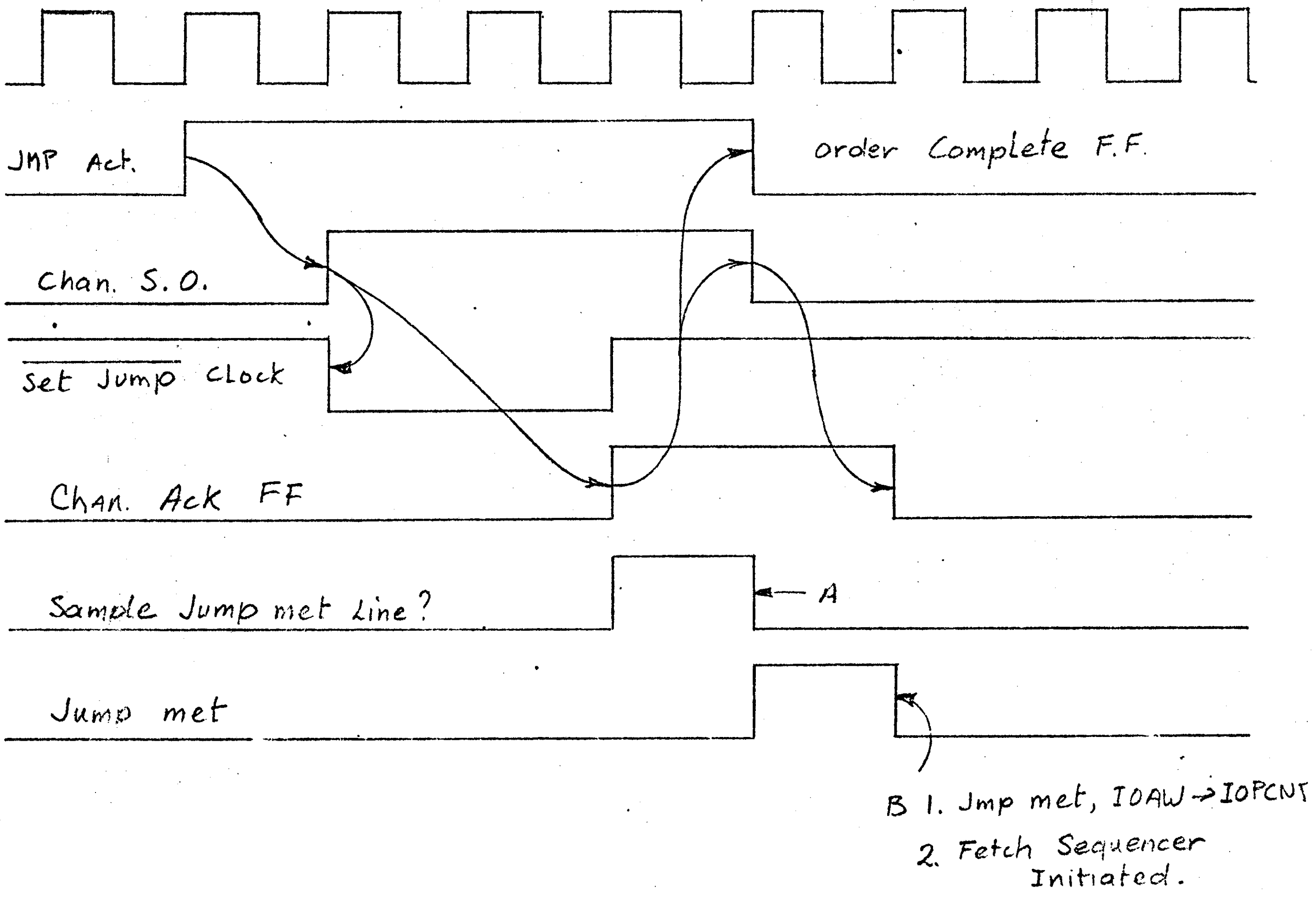


Figure 48. Conditional Jump Timing (HSC)

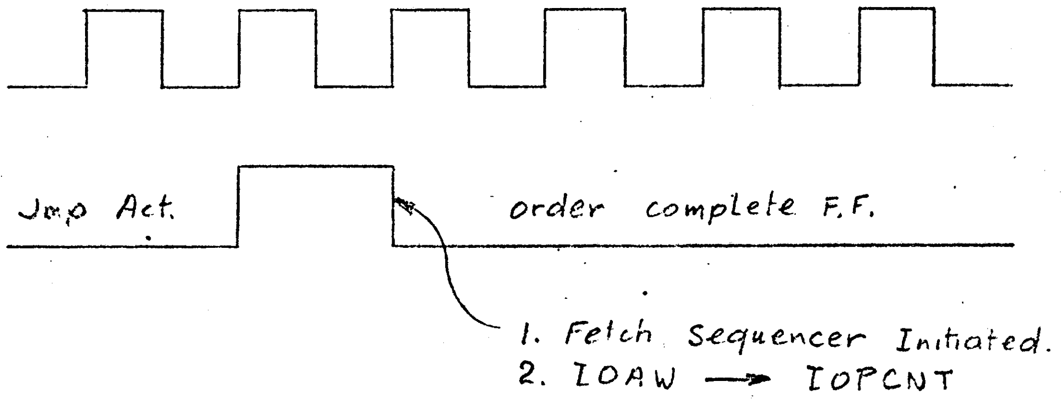


Figure 49. Unconditional Jump Timing (HSC)

controller. In both cases valid data is not present preceding the strobes, but is present 80 nsec following the trailing edge of the strobes.

9.7.7 Read and Write

These two orders have similar timing diagrams with two small exceptions. First, the Read order generates Read Next Word Strobes and the Write order does not. The second and most obvious difference is the direction of data transfer, Read, from the device controller and Write, to the device controller.

Figures 51 and 52 are the diagrams for Read and Write orders, the differences are pointed out, without chaining, and terminated both by DEV END and Word Count rolling over to 0. When terminated by Word Count rolling over to zero 0, it was assumed that initially the WC Register was set to -3.

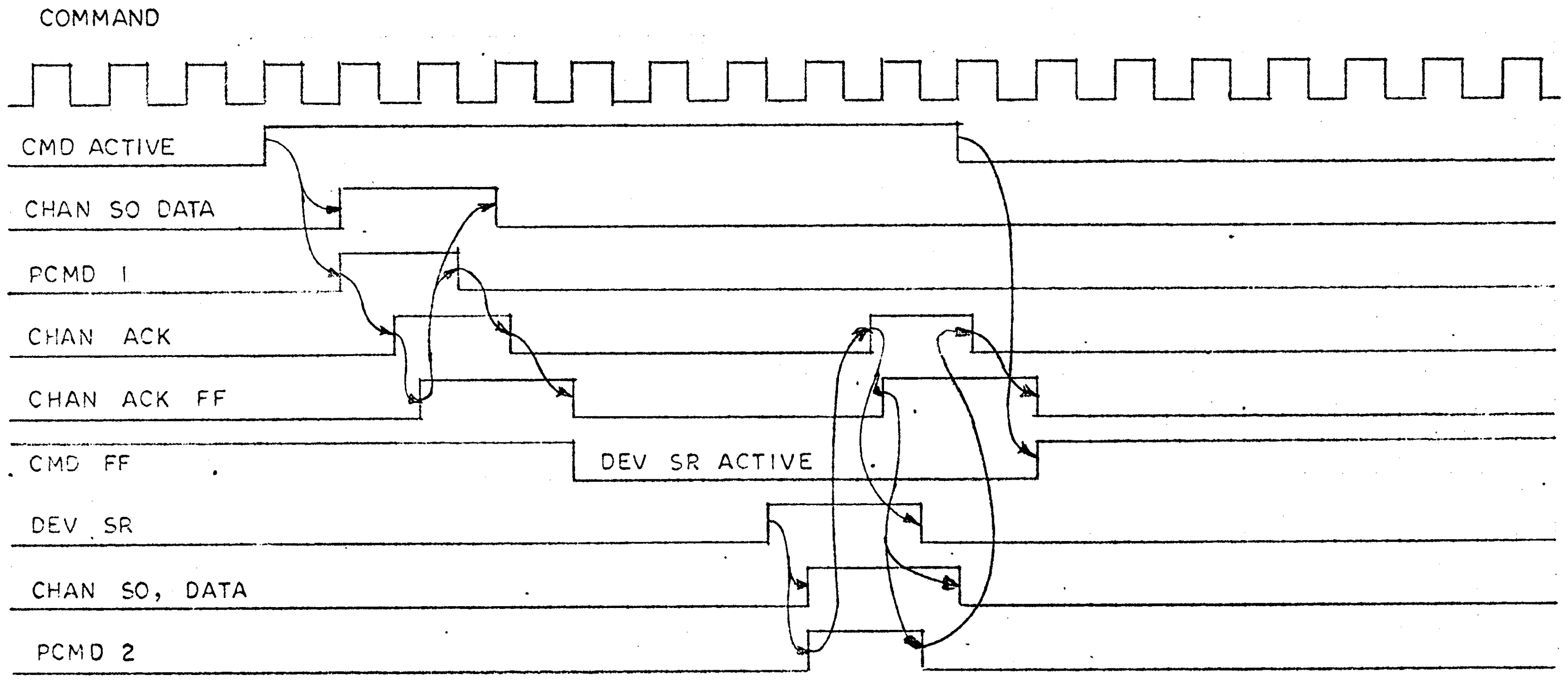
Read and Write orders differ somewhat in the way Transfer Cycles are initiated. During Reads and Writes, the Transfer Cycles are initiated by the buffer controllers. These controllers attempt to keep both buffers full during Write orders, and both buffers empty during Read orders. The other orders which transfer data, e.g. Sense, generate Transfer Cycles from the PEC. This is not the case for Read and Write Orders.

To fully understand the operation of DEV END, the reader should refer back to the previous section 9.2.3 where it is defined explicitly.

9.8 CHANNEL FLOWCHART

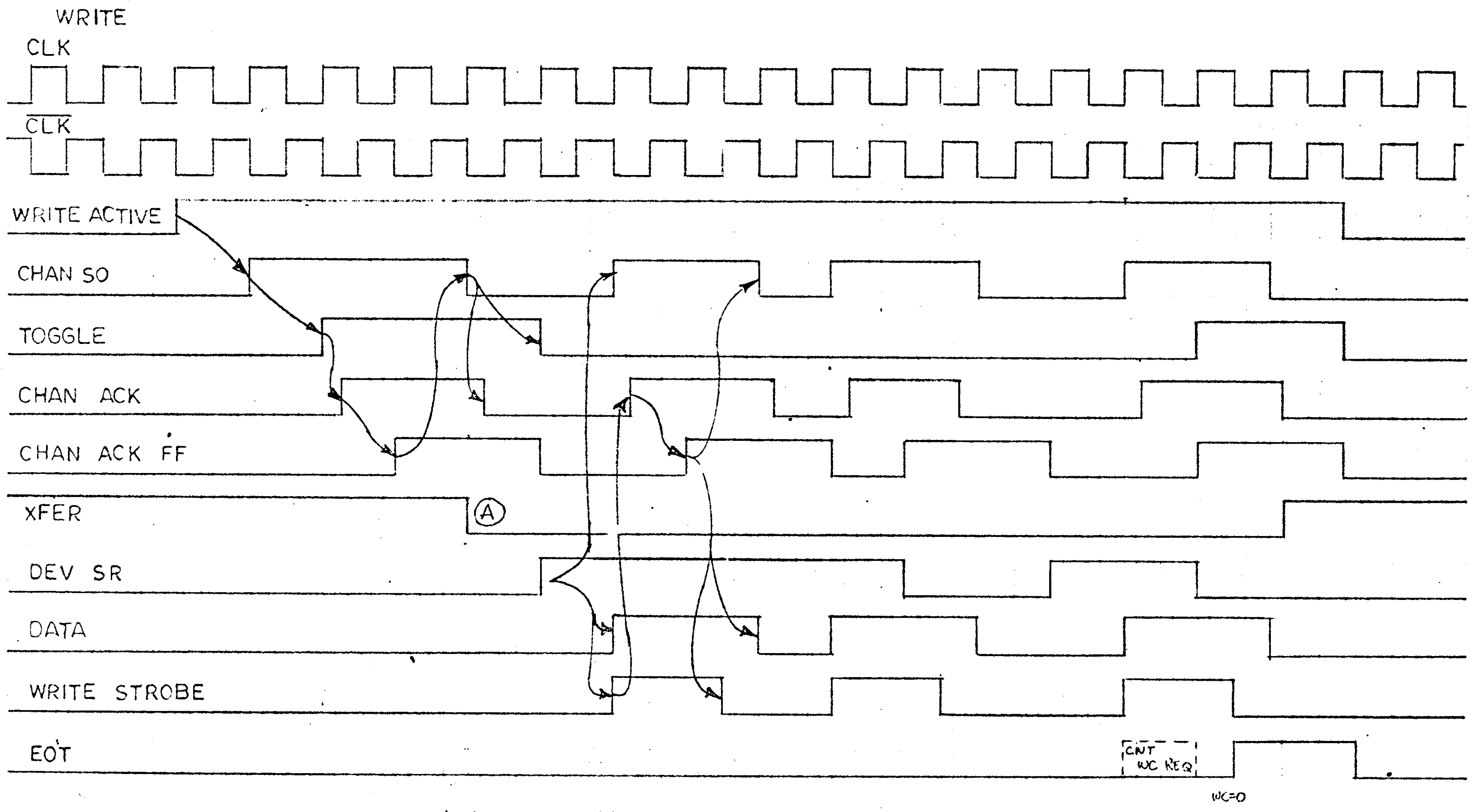
The overall Flow Chart for the Selector Channel appears in Figure 53. This chart shows the various controllers and sections of the logic discussed.

To help understand the chart the block WAIT needs to be defined. This block appears whenever a CHAN SO cycle is in progress. It is not shown for the Read and Write Orders. The charts pertaining to the Read and Write orders are the actual buffer controllers and not the Program Execution Controller.



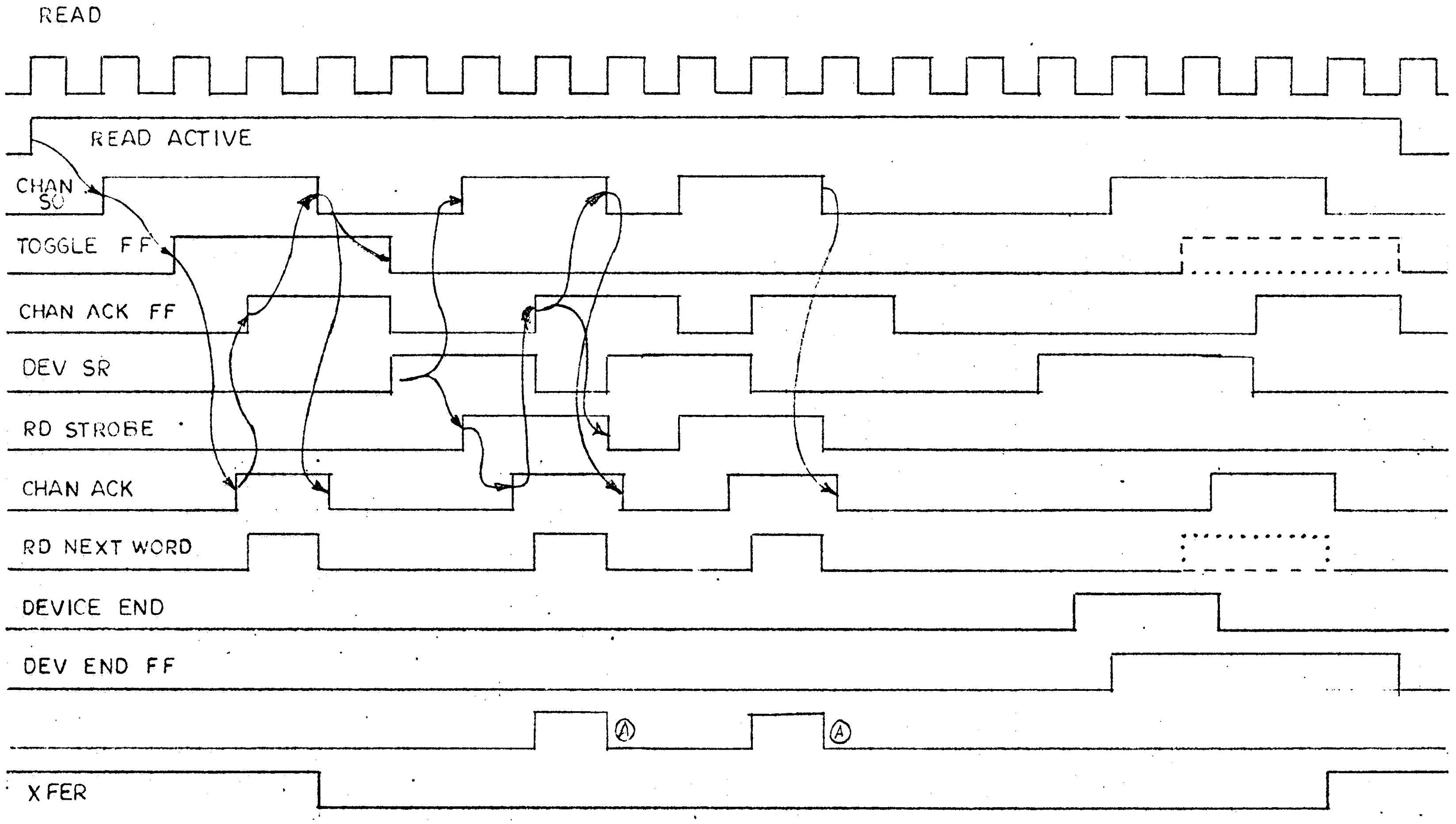
DATA IS PRESENT ON THE DATA LINES for the duration of Channel SERVICE OUT. IT IS NOT VALID AT THE LEADING EDGE of the Programmed STROBES

Figure 50. Command Timing (HSC)



Ⓐ DEV. SER. REQ. enabled into the HSC. THE HSC will not honor pending DEV SR for DATA UNTIL this time.

Figure 51. Write Timing (HSC)



Read with no chaining, when DEV END is recognized, the Read Strobe and Read next word Strobe are aborted.

Ⓐ Data is loaded into input buffers with this enable. This enable is generated by anding Read Strobe and Channel Ack.

– – – WITHOUT CHAIN
. . . . . . WITH CHAIN

Figure 52. Read Timing (HSC)

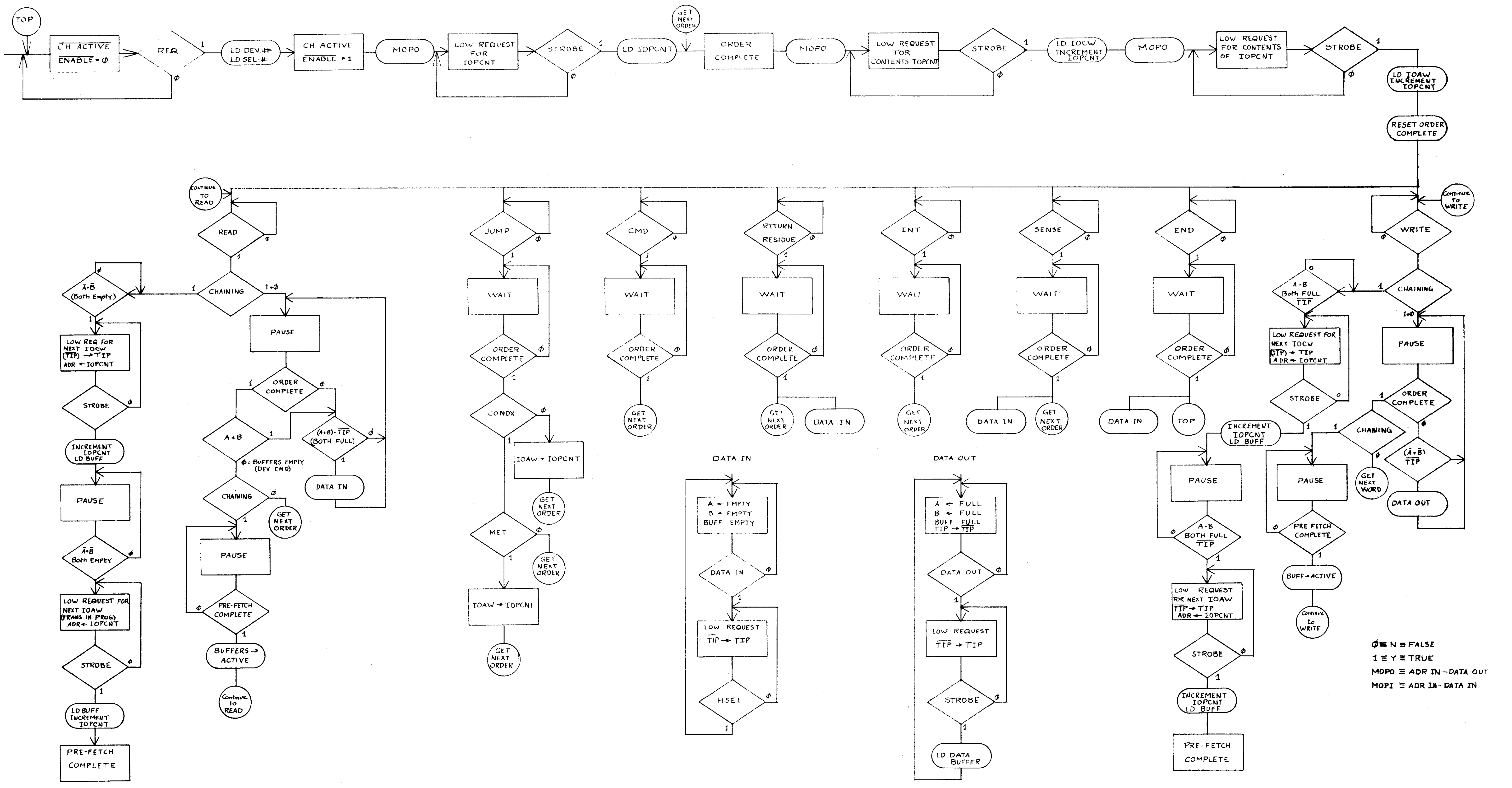


Figure 53. Overall Selector Channel Flow Chart

The two data buffers are indicated by A and B. The Write buffer controller attempts to keep both buffers empty.

Definitions for the flow chart in figure 53 are as follows:

A, B	-	Data Buffers
IOCW	-	IO Command Word
IOAW	-	IO Address Word or Operand
TIP	-	Transfer In Progress
$\overline{\text{TIP}}$	-	No transfer in Progress
WAIT	-	CHAN SO cycle in Progress
MOP	-	Memory operation
IOPCNT	-	IO Program Counter.
PAUSE	-	A STATE where the HSC is waiting for some operation to complete. The various pauses shown are not necessary the same STATE.

9.9 HSC INTERFACE

The interchange diagram, figure 54, shows all the signals in the Channel Bus. Each HSC has one bus for all its device controllers. The diagram also shows which logic, the HSC or the device controller, initiates the signal. Those signal names listed under the HSC are initiated by the HSC, those listed under the device controller are initiated by the device controller.

The 16 data lines are used by the device controller to transmit its device number and select number to the HSC during the Initiator sequence.

The REQ, CLR IL line is time shared, both signals initiating from the device controller.

The arrows indicate the signal direction. Double arrows indicate a bi-directional bus.

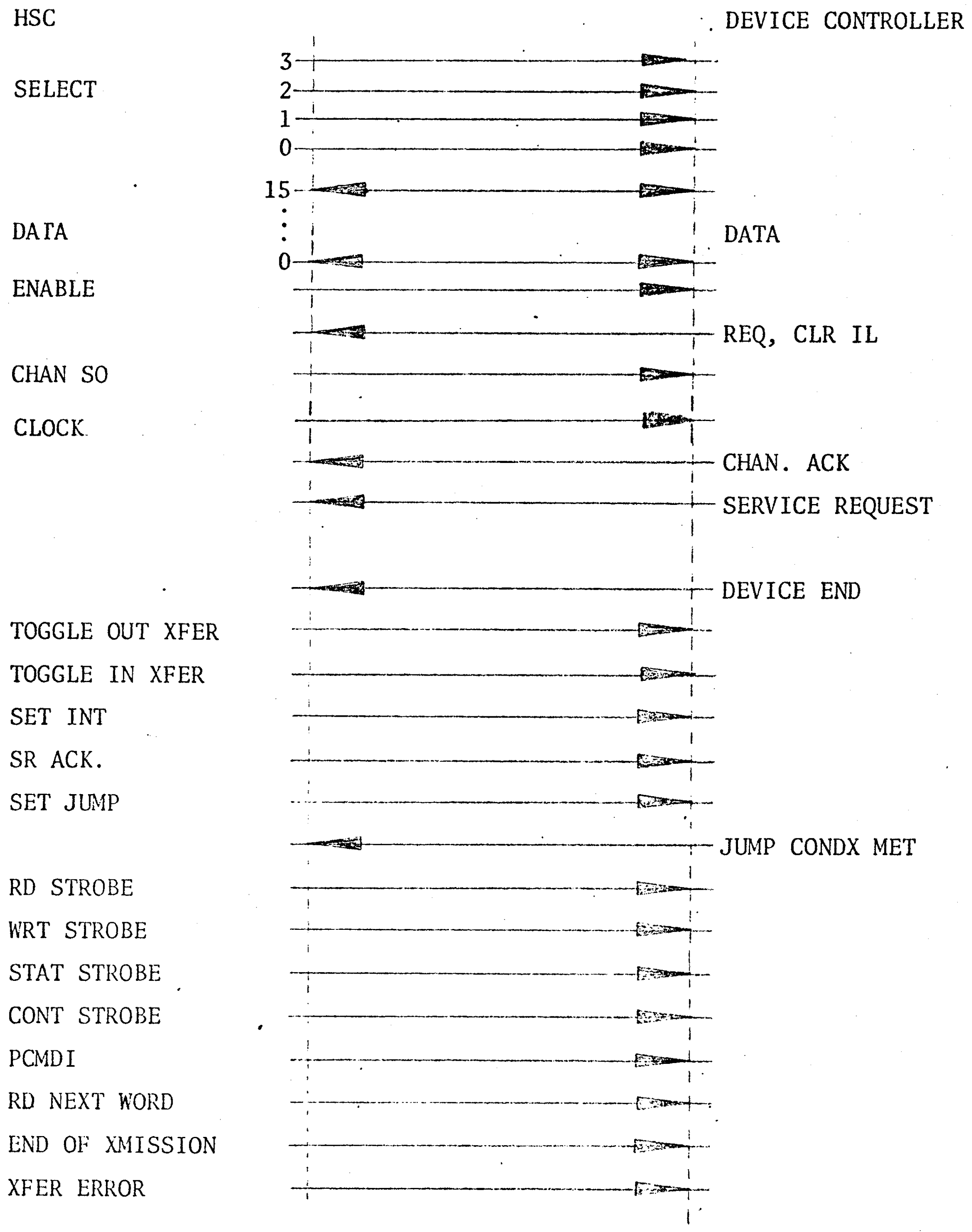


Figure 54. Channel Bus Signal Lines

9.10 PORT CONTROLLER BUS LINES

The control lines in the Port Controller bus are grouped into four sets. See figure 55. Those marked by #1 belong to one set, #2 to a second set, #3 to a third set, and #4 to the fourth set. Each set represents one HSC interface. The 16 data lines are common to all HSC along with the data parity line. Similarly system Clock, Master Reset and Channel Error line are common to all HSC.

Signal Definitions

TO: These lines represent the output of the address to module mapper in each HSC. The Port Controller interrogates the addressed module RDY condition before initiating a MCU Bus transfer.

MOP;LR: These lines perform two functions. Each Transfer cycle is initiated by the LR (Low Request) and the type of transfer to be initiated is coded by the MOP lines.

STROBE: Data directed to each HSC from memory in response to a Read Write memory cycle is "strobed" into the respective HSC by the STROBE lines.

LSEL: The Port Controller strobes the address from the respective HSC onto the MCU bus with the LSEL signal.

HSEL: The Port Controller strobes the data from the respective HSC onto the MCU bus during a clear Write memory cycle with the HSEL signal.

Channel Error Line: The Port Controller passes the state of the Address Error Line and the System Parity Error logic onto the HSC Error Controllers.

The sets of control lines are jumper selectable in the individual HSC. Each channel must be a different number 1 to 4.

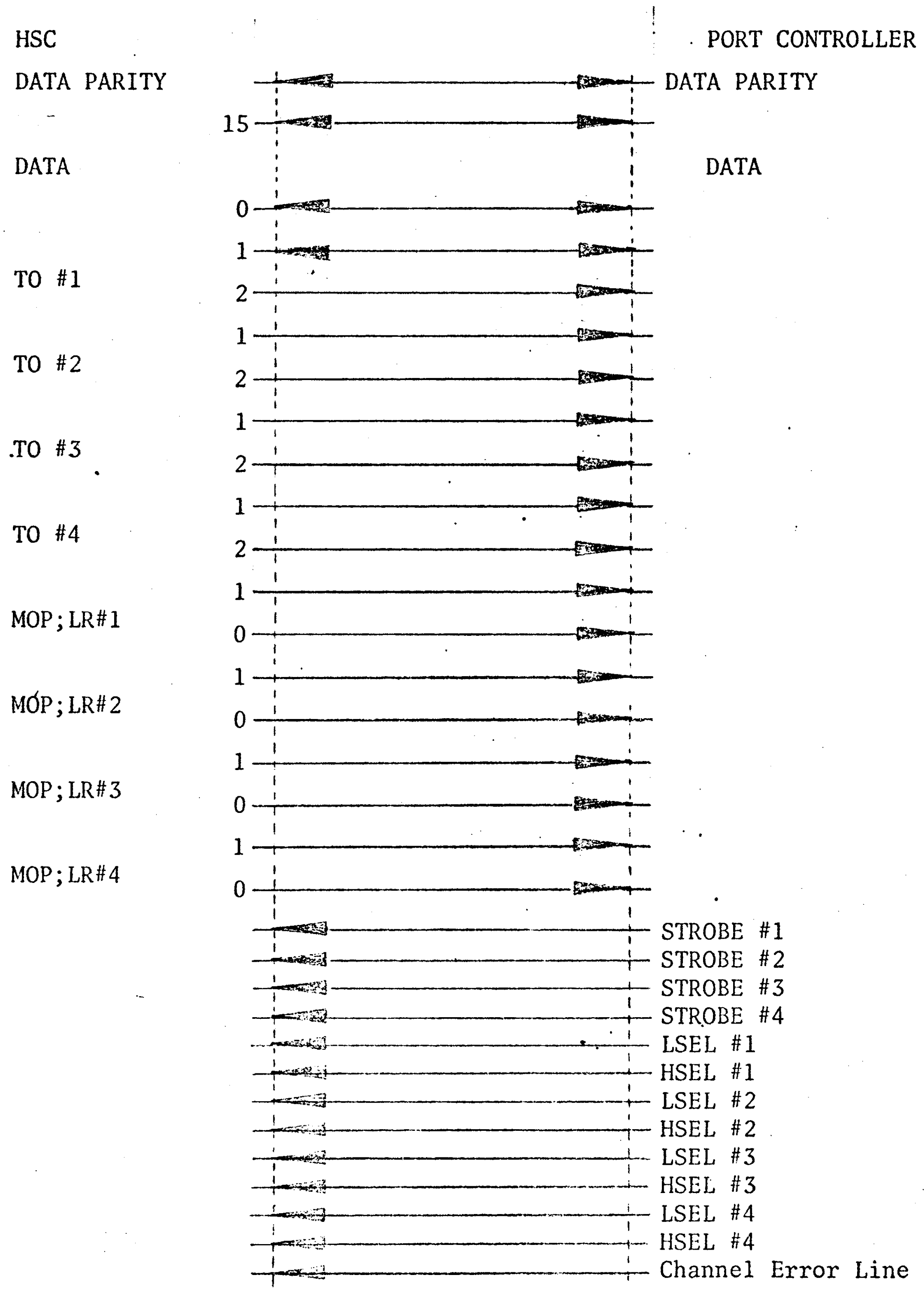


Figure 55. Port Controller Bus Signal Lines

9.11 SERVICE REQUEST FLIP-FLOP

Figure 56 shows a typical device SE FF showing the clock and a method for resetting the FF.

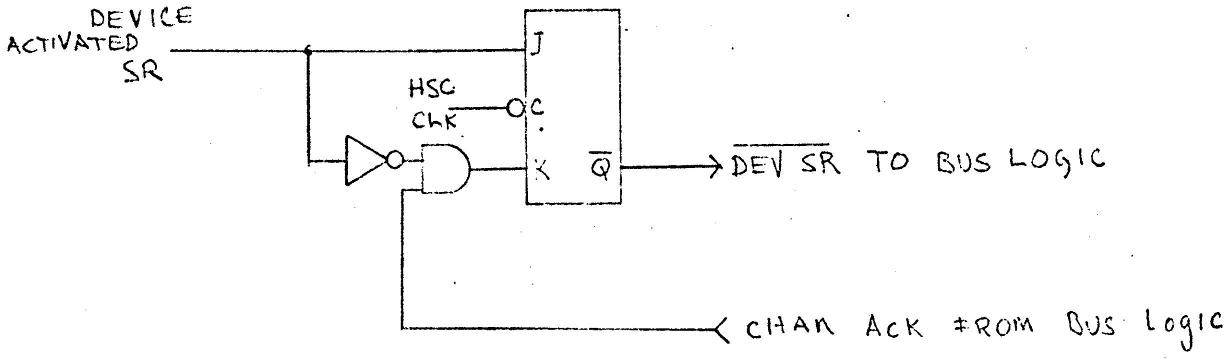


Figure 56. Service Request Gating (HSC)

SECTION 10

COLD-LOAD OPERATION

10.1 INTRODUCTION

The Cold-Load feature is intended to replace the usual core-resident bootstrap loader normally required to load a program into memory when a system is initiated. The Cold-Load feature is even more valuable if volatile memories are used, and the bootstrap loader is destroyed when power is shut off.

The operation consists of the microcode initiating a transfer from a device into core according to bits specified in the switch register when the Cold-Load button is pushed. The first data transferred by this means is then used as an I/O program to continue the transfers from the device to core at addresses specified within the first block of data transferred. When the transfers that have been specified are complete, the device issues an interrupt which can start the CPU processing at a location that is specified by the data transferred from the device.

10.2 COLD-LOAD SPECIFICATION

When the COLD-LOAD button is pushed, the processor reads bits (8:15) from the switch register, and creates the following I/O program at location (device number *4):

*+1	DRT I/O Program counter for device
%40000$_8$	I/O Control Command
SW(0:7) right just	8-bit Control Byte from (0:7) of SW. Reg.
Read -%40$_8$	I/O Read command of %40 words
*+1	Read address of *+1

At this point the CPU starts the I/O program shown above running. The device receives a control word, specified by SW (0:7), followed by a read of %40$_8$ words into the I/O program. Then the I/O program continues by executing the first words transferred in by the read of $\%40_8$. These words are expected to cause blocks to be transferred in that will establish the stack, code domain, and I/O drivers for whatever program that is being COLD-LOADed.

The CPU, after initiating the I/O program waits for an external interrupt from the COLD-LOAD device with registers set to the following values:

PB	=	0
PL	=	$\%177777_8$
P	=	contents of word 1 of memory

This establishes a code domain of a starting address of the location specified by word 1 of memory. It is assumed that this cell was set up by one of the transfers specified by the 40 word block first brought in.

The stack domain is set up as follows:

DB	=	0
Q	=	Qi
Z	=	Zi
DL	=	0
S	=	Q + 1
Interrupt Stack Flag	=	1

This is the normal interrupt stack environment. The STATUS register is set to $\%140006_8$. This indicates privileged mode, interrupts enabled, and segment 6.

10.3 COLD-LOAD REQUIREMENTS

For devices such as disks, where it is expected that COLD-LOAD will be used to start up operating systems, it is important that the controller be capable of doing this with the format of the I/O program specified above. For example, the 8-bits of control information can be used to specify a track number that the transfer will be made from.

Also in the case of a disk, it is advisable that the MASTER CLEAR signal reset the cylinder, track, and sector registers to zero so that when the control word to start a COLD-LOAD operation at the specified location occurs, the transfer will begin at sector 0.

Note that the first words read in from the device are executed as I/O program. This means that if the device should return with zeros or bad data, then the COLD-LOAD will be hopelessly lost. For example, in a paper tape reader, if feed holes are read, then the zeros will cause the COLD-LOAD to crash. A possible solution that could work would be to have a control word which would pass up feed holes in tape.

Of course, it is always possible to require that the reader be aligned to the first data word on the tape before the COLD-LOAD button is pushed.

SIO BUS (P2)

1.	$\overline{\text{CHAN SO}}$	23.	GND	45.	$\overline{\text{P STATUS STB}}$
2.	GND	24.	$\overline{\text{SR-DATA 15}}$	46.	$\overline{\text{P CONT STB}}$
3.	CLOCK	25.	$\overline{\text{SR-DATA 14}}$	47.	$\overline{\text{Rd NEXT WD}}$
4.	GND	26.	$\overline{\text{SR-DATA 13}}$	48.	$\overline{\text{P WRITE STBE}}$
5.	$\overline{\text{DEV END}}$	27.	$\overline{\text{SR-DATA 12}}$	49.	$\overline{\text{SET INT}}$
6.	GND	28.	$\overline{\text{SR-DATA 11}}$	50.	$\overline{\text{P READ STB}}$
7.	$\overline{\text{ACK SR}}$	29.	$\overline{\text{SR-DATA 1Ø}}$		
8.	GND	30.	GND		
9.	$\overline{\text{CHAN ACK}}$	31.	$\overline{\text{SR-DATA 9}}$		
10.	GND	32.	$\overline{\text{SR-DATA 8}}$		
11.	$\overline{\text{DEV \# } \rightarrow \text{ DB}}$	33.	$\overline{\text{SR-DATA 7}}$		
12.	$\overline{\text{ENABLE}}$	34.	$\overline{\text{SR-DATA 6}}$		
13.	$\overline{\text{EOT}}$	35.	$\overline{\text{SR-DATA 5}}$		
14.	$\overline{\text{JMP MET}}$	36.	GND		
15.	GND	37.	$\overline{\text{SR-DATA 4}}$		
16.	Toggle INXFER	38.	$\overline{\text{SR-DATA 3}}$		
17.	$\overline{\text{TOGGLE SR}}$	39.	$\overline{\text{SR-DATA 2}}$		
18.	Toggle OUTXFER	40.	$\overline{\text{SR-DATA 1}}$		
19.	Toggle SIO OK	41.	$\overline{\text{SR-DATA Ø}}$		
20.	GND	42.	GND		
21.	$\overline{\text{XFER ERROR}}$	43.	$\overline{\text{P CMD 1}}$		
22.	$\overline{\text{REQ}}$	44.	$\overline{\text{SET JMP}}$		

I/O Alpha Pin Assignments (P3)

*1.	$\overline{\text{PARITY}}$
2.	$\overline{\text{PARITY ERROR}}$
3.	GND
*4.	$\overline{\text{CMD IN-OUT Ø}}$
*5.	$\overline{\text{CMD IN-OUT 2}}$
*6.	$\overline{\text{CMD IN-OUT 1}}$
7.	GND
*8.	$\overline{\text{DEV-INT ADR Ø}}$
*9.	$\overline{\text{DEV-INT ADR 1}}$
10.	GND
*11.	$\overline{\text{DEV-INT ADR 2}}$
*12.	$\overline{\text{DEV-INT ADR 3}}$
13.	GND
*14.	$\overline{\text{DEV-INT ADR 4}}$
*15.	$\overline{\text{DEV-INT ADR 5}}$
*16.	GND
*17.	$\overline{\text{DEV-INT ADR 6}}$
*18.	$\overline{\text{DEV-INT ADR 7}}$
19.	GND
*20.	$\overline{\text{DATA Ø}}$
*21.	$\overline{\text{DATA 1}}$
22.	GND
*23.	$\overline{\text{DATA 2}}$
*24.	$\overline{\text{DATA 3}}$
25.	GND
*26.	$\overline{\text{DATA 4}}$
*27.	$\overline{\text{DATA 5}}$
28.	GND
*29.	$\overline{\text{DATA 6}}$
*30.	$\overline{\text{DATA 7}}$
31.	GND
*32.	$\overline{\text{DATA 8}}$
*33.	$\overline{\text{DATA 9}}$
34.	GND
*35.	$\overline{\text{DATA 10}}$
*36.	$\overline{\text{DATA 11}}$
37.	GND
*38.	$\overline{\text{DATA 12}}$
*39.	$\overline{\text{DATA 13}}$
40.	GND
*41.	$\overline{\text{DATA 14}}$
*42.	$\overline{\text{DATA 15}}$
43.	GND
44.	$\overline{\text{INT REQ 1}}$
45.	$\overline{\text{INT REQ 2}}$
46.	GND
47.	$\overline{\text{INT REQ 3}}$
48.	$\overline{\text{INT REQ 4}}$
49.	GND
50.	$\overline{\text{INT ACK}}$

*≡Bi-directional (IOP Requires termination on all lines except No. 2)