CSP-007

SIGMA SERIES INPUT/OUTPUT PROCESSORS

Prepared for

SDS Data Systems

Beverly Hills, California

10 March 1966

CONTENTS

FIGURES

SECTION I: GENERAL

1.0 DIFFERENT TYPES OF INPUT/OUTPUT

1.1 RWD Interface

The Read-Write-Direct interface gets its name from the two SIGMA instructions that control this interface: Read Direct (RDD), and Write Direct (WRD). This interface is similar in a sense to the POT/PIN interface on the SDS 900 Series computers; RDD is analogous to an EOM/PIN sequence of instructions, while WRD is analogous to an EOM/POT sequence. There is no analogy for the SKS instruction. This function is accomplished in another way, described in Section I-1.4.

Each instruction (RDD or WRD) presents 16 bits of effective address at the interface, along with a control signal designated Function Strobe. The external unit must provide a signal called Function Strobe Acknowledge which, like $\overline{Rf}$ in the 900 Series computers, allows the RDD or WRD instruction to be executed.

During RDD, the external unit may provide 32 bits of data for storage in a computer register; this data is provided in conjunction with Function Strobe Acknowledge. During WRD, the computer provides 32 bits of data to the external unit (in conjunction with Function Strobe) on the same 32 lines that are used to transmit input data during RDD. Thus the 32 data lines are time-shared for the two different instructions.

Additional signals provided at the interface are:

- a line that differentiates between RDD and WRD for the external user (since they appear identical in all other respects),
- two additional data lines for input (they are input lines even during WRD), which set the computer's condition code bits 3 and 4,
- an I/O Reset signal, and
- a 1-Mc clock signal (not synchronized in any way with instruction execution.

On SIGMA 2, the RWD interface is identical, except that the data path is only 16 bits wide. Thus, only three cables are required for SIGMA 2, rather than the four cables on SIGMA 7 and SIGMA 5, since each SIGMA cable contains 14 conductors.

1.2 Eight-Bit Data Path

The eight-bit data path, referred to as the Input/Output Processor (IOP) interface, is analogous to the TMCC/DACC interfaces in the SDS 900 Series computers. One eight-bit data path is associated with each IOP in a system (there may be up to eight IOP's). Each path is logically and electrically independent of every other path, in every respect. Device Controllers (peripheral control units) are attached to this interface in a completely parallel manner, requiring only one set of four cables to be connected to the IOP; each Device Controller is connected to the one before it (nearer physically to the IOP). Section I-3 gives a complete description of the electrical characteristics of this interface, and Section II-3 describes the cabling.

Up to 32 Device Controllers may be attached to any IOP interface. All Device Controllers time-share the single eight-bit data path. Operation resembles interlaced operation on the 900 Series, except that the interlace word count and each main memory address are stored in an IOP private fast memory for each Device Controller attached to the bus. Since the private fast memory also contains other control information associated with each Device Controller, 56 bits of storage in the IOP are assigned exclusively to each possible device address. Each device connects to the IOP through a Device Controller (DC).

Once a device has been "started" by the main computer program, the general scheme of I/O interaction is as follows:

1. Any number of Device Controllers may request "service" simultaneously from the IOP.
2. The IOP brings up certain signals that activate a hard-wired priority chain between Device Controllers. The highest priority Device Controller that has requested service puts its own device address on some return lines, along with an acknowledge signal, and is then "connected" to the IOP for a short time.

3. While it is connected to the IOP, a Device Controller makes its request: Data Out, Data In, control information from the CPU memory, or transfer of control information into the IOP fast memory.

4. While it is connected to an IOP, a Device Controller can transmit or receive up to four (8-bit) bytes of data or one byte of control information, plus, in some cases, an additional byte of control information known as a Terminal Order.

It should be noted that up to 32 devices on a given IOP may request service simultaneously. They are serviced in sequence by the IOP, based on the hard-wired priority chain. The brief description above is intended to serve only as an introduction; Section III describes the IOP interface in greater detail.

This interface is compatible on SIGMA 2, SIGMA 5, and SIGMA 7, except for some differences in bit coding for certain exchanges of control information and in the handling of Order In, Order Out, and Terminal Orders. These differences are discussed in III-4.

The different types of IOP's that may be utilized with a SIGMA system include:

M/IOP	Multiplexor IOP (basic unit)
S/IOP	Selector IOP
S'/IOP	Selector Piggyback IOP

Throughout this manual, the letters IOP generally refer to the M/IOP interface.

All IOP's present the same interface and general signal interaction to a Device Controller. Any device designed to operate on the IOP interface described in this manual may connect to any of the specified types of IOP's. However, the S/IOP is designed to allow much higher data rates (up to 1.5 million bytes per second) versus a maximum of approximately 0.3 million for an M/IOP. Also, the S/IOP permits only one device to be active at any given time, as compared to up to 32 active devices for the M/IOP.

The S'/IOP is an additional S/IOP that shares the same bus as its parent S/IOP. It is identical in all other respects. The S/IOP and S'/IOP also permit Device Controllers that operate with a 32-bit data interface, whose control signals are identical with the M/IOP. Obviously, such a Device Controller cannot be used on the 8-bit M/IOP.

Due to cabling and port access restrictions, the following formulas apply to system IOP configurations:

$$M + S \leq 5$$

$$M + S + S' \leq 8$$

where M = number of M/IOP's
S = number of S/IOP's
S' = number of S'/IOP's

1.3 Memory Bus

The IOP itself connects to the computer, as well as to the memory bus for some memory module. Although it is seldom necessary for system requirements, an external device can connect directly to this memory bus.

The memory bus comprises 32 data lines (time-shared for input and output), 17 address lines, a Memory Request signal, a Data Release line, an Address Release line, an Address Here signal, and a Parity Error signal. There are also four Write Byte control lines that are used during partial write operations to specify which of the four bytes in a memory cell are to be altered during that operation.

The memory bus is described in detail in Section IV.

1.4 Testing of External Signals

SIGMA systems do not have any explicit instructions to allow external signals to be tested through the RWD interface, analogous to the SKS instruction in 900 Series computers. However, this function can be accomplished quite readily by means of an RDD instruction (which stores up to 32 bits in a specified register), followed by a Compare Selective (CS) instruction, and then a test of the condition code bits (whose settings are based on the results of the CS). Single or multiple bits may be tested by this procedure. The SIGMA 5 and SIGMA 7 reference manuals describe how these instructions operate.

2.0 SIGMA COMPATIBILITY

SIGMA 5 and SIGMA 7 are completely identical with respect to all interfaces. SIGMA 2 has certain differences on the various interfaces.

On the RWD interface, the only difference of consequence to the external-unit designer is the word size: 16 bits on SIGMA 2 versus 32 bits on SIGMA 5 and SIGMA 7. The means to accommodate this difference for standard products are discussed in Section II-5.

On the IOP interface, it is an unbreakable rule that all units must be capable of operating equally well on SIGMA 7, SIGMA 5, or SIGMA 2. This will affect design with respect to the means of identifying certain conditions. For instance, on SIGMA 5 and SIGMA 7, Terminal Orders always follow an Order Out cycle; on SIGMA 2 they may not. Thus, the logic conditions that cause a Terminal Order state must be derived from identifying signals provided by the computer, rather than directly from the occurrence of the Order Out cycle itself. This issue is discussed in Section III, and particularly in Section III-4.

3.0 ELECTRICAL DESCRIPTION OF INTERFACES

All interfaces – RWD, IOP, and memory bus – utilize the same cable driver/receiver scheme, and the same means of inter-unit cabling. All inter-unit cabling consists of several cables. Each cable consists of 14 shielded wires, whose characteristics are given in Appendix A. Each shielded wire is used as a transmission line which is terminated with its 33-ohm characteristic impedance at each end. Connection to this line is by virtually tapping the line without unduly affecting its characteristic impedance. A number of cables are used to handle all signals associated with each interface. These are:

• RWD Interface	4 cables for SIGMA 5 and SIGMA 7 3 cables for SIGMA 2
• IOP Interface to Device Controller	3 cables for all (There is, in addition, a single priority-determination cable running only between Device Controllers; it does not connect to the IOP.)
• IOP Interface to Memory	5 cables for all
• IOP Interface to Computer	1 cable for all
• Memory Bus	5 cables for all

Figure 1-1 on the following page shows the cabling scheme.

For all signals except those associated with priority determination among Device Controllers, each unit may tap any line with both a driver and a receiver. The following conventions have been adopted:

• Logic One	+2 volts	driver output = low impedance
• Logic Zero	0 volts	driver output = high impedance

It should be noted that this reverses the conventions in the SDS 900 Series equipment, where a logical one was the high impedance state and logical zero was the low impedance state. Thus, the quiescent state of time-shared lines must be altered: the

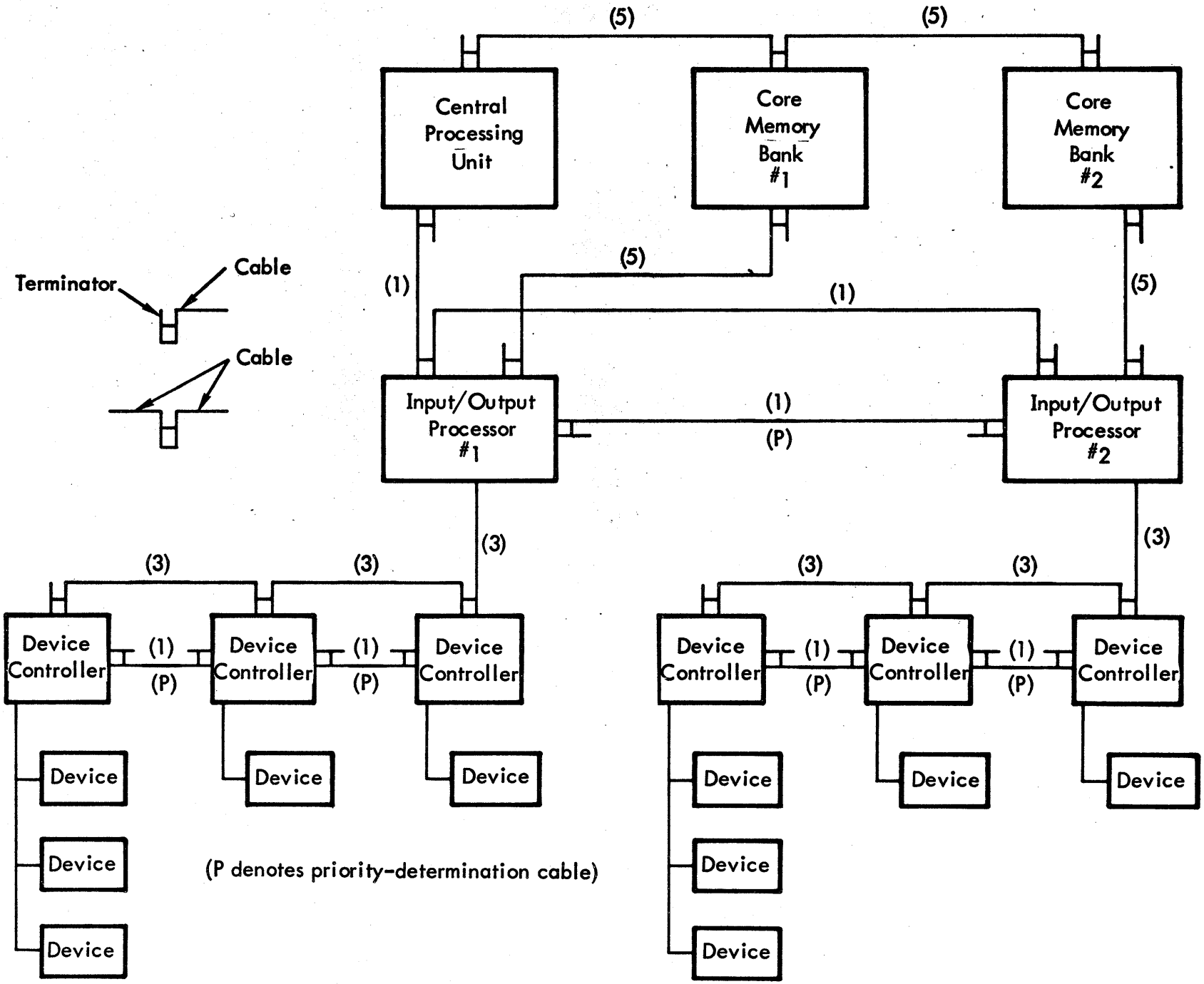


Figure 1-1. SIGMA 7 Cabling Schematic

quiescent state of any time-shared line must be high impedance from the driver (logical zero). Any single unit whose driver is active (turned on) on a given line may bring that line to the logical-one state. This is accomplished by the driver circuit shown in Figure 1-2 on the following page.

Receiver circuits similarly tap the common line for each signal; they consist of high-noise-rejection discriminator circuits.

There is no inversion in either drivers or receivers. If the input to a driver is a logical one (+4 volts), the line driven goes to logical one (+2 volts). If a line is at logical one (+2 volts), the receiver output is at logical one (+4 volts).

The priority determination signals are carried by exactly the same sort of scheme, except that normally only one driver and one receiver are attached to each line. Signals are received by each unit, passed through decision-making ordinary logic elements, and then they may or may not be passed on to the next unit.

Except for length, inter-unit cabling is all physically identical. There are no special components or circuits on the cables themselves. Cables are attached to the back of cable driver/receiver modules (AT10, AT11, AT12, AT17). These modules may be plugged into any slot on a standard SIGMA chassis; however, there are standard slots, for standard interfaces, into which they should be wired so they can be identified easily by Customer Service personnel.

Each such module can accommodate two cables, which are bussed (by etching on the module) directly to each other, as well as to the active circuits on the module (drivers and/or receivers). Thus, inter-unit cabling comprises one long transmission line, with drivers and receivers attached at each unit. The "last" unit on the transmission line (at both ends) must have a terminating network attached to the line in place of the cable that would ordinarily run out from that point, as shown in Figure 1-3.

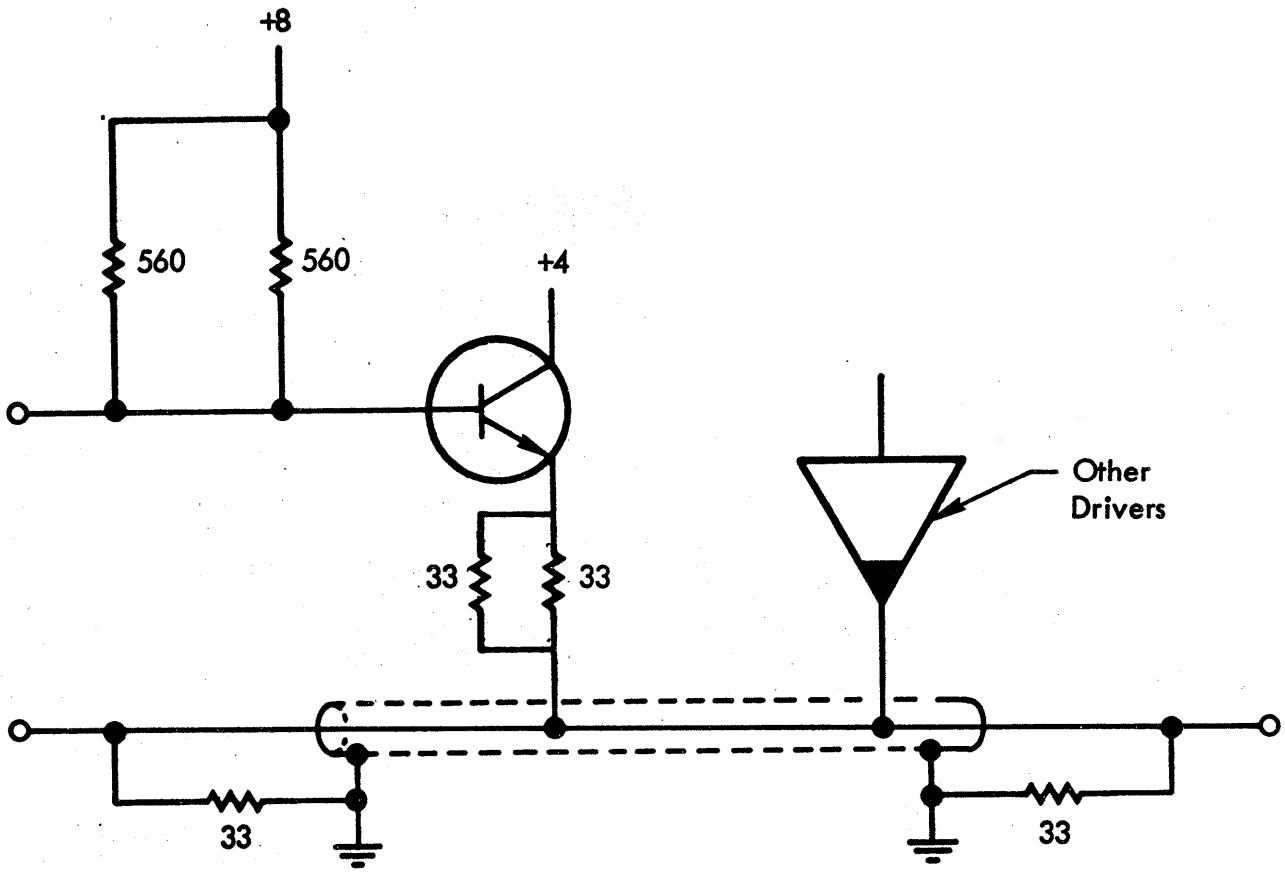


Figure 1-2. Driver Circuit Schematic

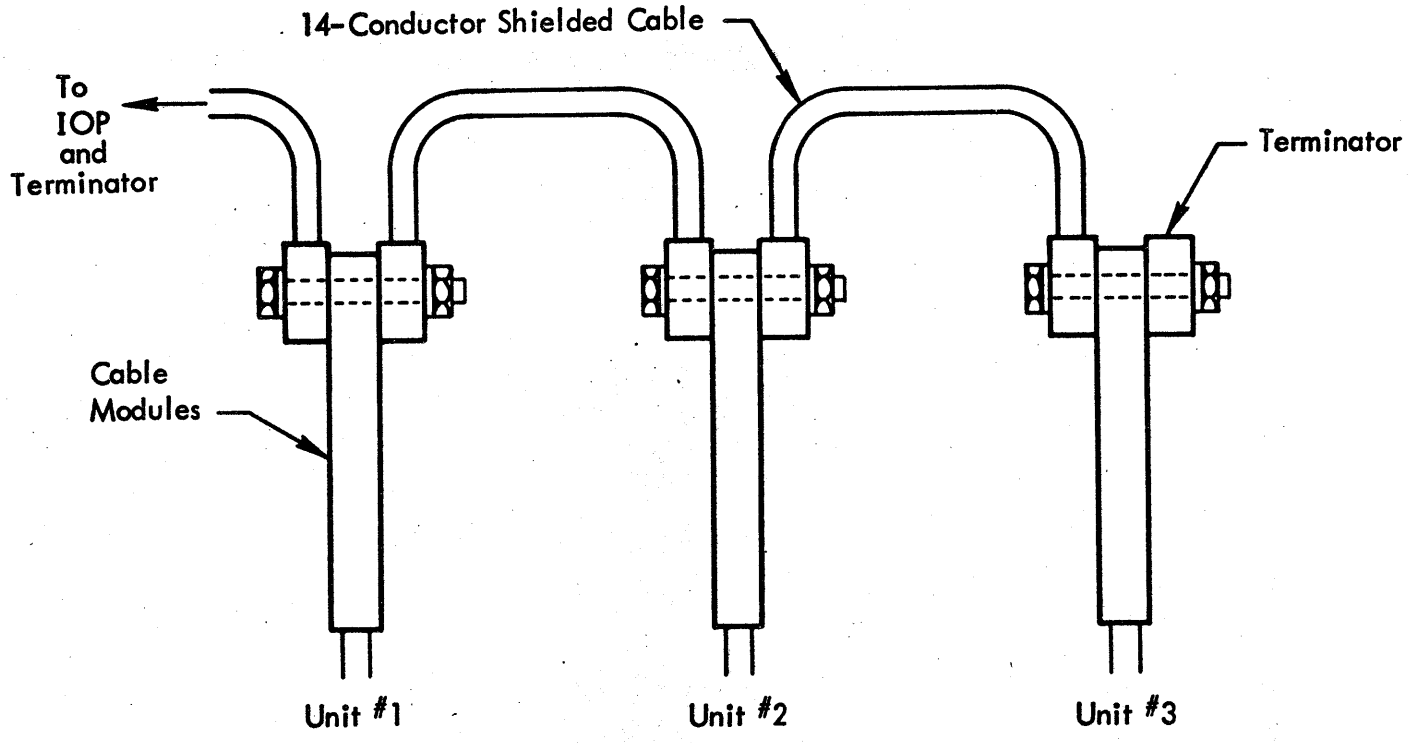


Figure 1-3. Cable and Terminator Connections

4.0 EXPANSION

The philosophy of design for the SIGMA input/output system dictates that elements in the system should be separable and easily connected. Therefore the following design goals should be noted carefully:

- All cables should be identical except for length.
- Cable receptacle locations should occur in the same place within each chassis assembly.
- Similar assemblies should be connected in similar fashion.
- Cables are trunk to tail, and there should be no ambiguity as to the means by which interconnection or expansion of assemblies is mechanized.

Figure 1-4 on the following page exemplifies these ideas and demonstrates how expansion is physically realizable.

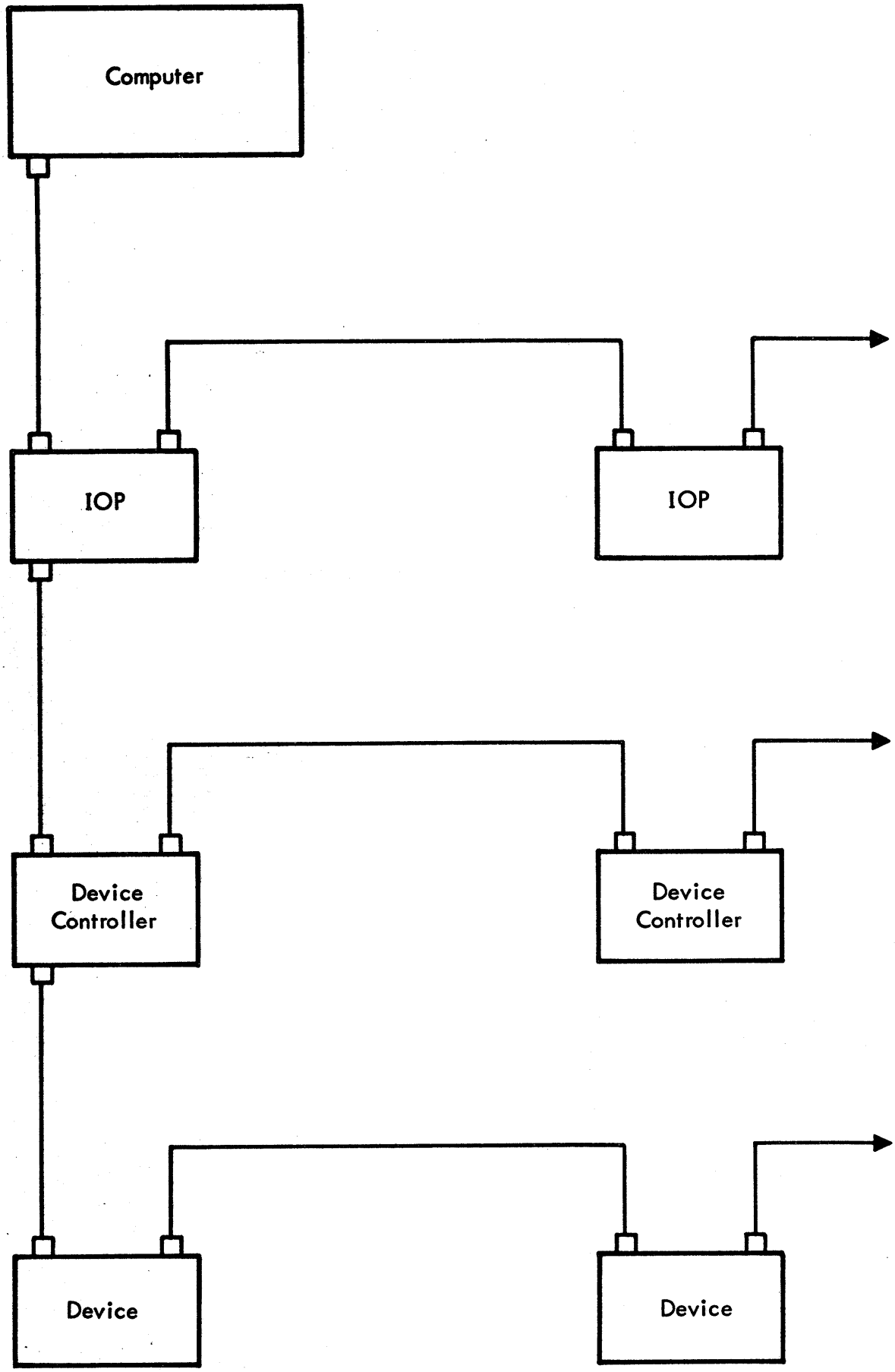


Figure 1-4. SIGMA Expansion

SECTION II: READ-WRITE-DIRECT INTERFACE

1.0 INSTRUCTIONS IN THE COMPUTER

1.1 Data Input

In SIGMA 5 and SIGMA 7 the Read Direct (RDD) instruction presents the low-order 16 bits of its effective address on the RWD interface address lines, along with a signal that indicates the execution of the RDD instruction. The selected external device must generate an acknowledge signal and may present up to 32 bits of data on the RWD interface data lines, as well as two bits of condition code information on two other lines to set CC3 and CC4. If R (the register-specifying field in the instruction) is not equal to zero, the 32 data bits are stored in the specified register. If R equals zero, the 32 data lines are ignored, but CC3 and CC4 are still set as specified by the external device.

In SIGMA 2 Read Direct (RD) is the same as RDD in SIGMA 5 and SIGMA 7, except that only 16 bits of data are accepted, and they are stored in the SIGMA 2 accumulator. The condition code input lines are stored in the SIGMA 2 carry and overflow indicators.

1.2 Data Output

In SIGMA 5 and SIGMA 7 Write Direct (WRD) presents the low-order 16 bits of its effective address on the RWD interface address lines, along with a signal that indicates the execution of the WRD instruction. The selected external device must generate an acknowledge signal, and may accept 32 bits of data from the RWD interface data lines; the computer simultaneously accepts two bits of condition code information on two other lines to set CC3 and CC4. If R (the register-specifying field in the instruction) is not equal to zero, the 32 bits of the specified register are transmitted on the 32 data lines. If R equals zero, the 32 data lines are set at logical zero regardless of the contents of any register, but CC3 and CC4 are still set as specified by the external device.

On SIGMA 2 Write Direct (WD) is the same as WRD for SIGMA 5 and SIGMA 7, except that only 16 bits of data are transmitted; they are taken from the SIGMA 2 accumulator. The condition code input lines are stored in the carry and overflow indicators in SIGMA 2.

1.3 Effective Address Assignments

The general area that is controlled by any RDD, WRD, RD, or WD instruction is specified by the high-order four bits of the effective address, and is designated as the control mode. The effective address bits are designated numerically as 16-31 in SIGMA 5 and SIGMA 7. Control mode assignments have been made as follows:

Hexadecimal	Definition
0	Internal Computer Control
1	Interrupt Control
2-B	Presently Unassigned
C-E	Systems Standard Products
F	Systems Special Units

A list of effective address assignments for control modes C to E is to be maintained by the Systems Engineering secretary. If a unit which might be applicable to future systems is to be designed, approval must be obtained from the Director of Systems Engineering for assignment of a standard WRD effective address coding. No central record will be kept for effective address assignments with control mode F.

For all standard product effective address assignments, the reservation of a particular code will apply to both SIGMA 5/SIGMA 7 and SIGMA 2, even if the standard product is designed to operate primarily on only one of the systems. It should be emphasized, however, that standard products should be designed, whenever possible, with a view towards effective operation on any SIGMA computer. Suggested means of accomplishing this are discussed in Section II-5.

2.0 INTERFACE SIGNAL LINES AND CABLING

The following cables are involved in the RWD interface:

Cable No.	Connects to	Transmits
1	AT11 cable driver/receiver module (23S in CPU)	data input/output bits 00-13
2	AT11 cable driver/receiver module (27Q in CPU)	data input/output bits 14-27
3	AT11 cable driver/receiver module (30P in CPU)	data input/output bits 28-31; Function Strobe; Function Strobe Acknowledge; RDD/WRD; and Address Lines 5-7 and 12-15
4	AT10 cable receiver module (29N in CPU)	CC3 and CC4; I/O Reset; 1-Mc Clock; Address Lines 0-4 and 8-11; and One Spare.

The following items should be noted:

- Cable No. 1 is not needed for a SIGMA 2 system.
- The designer must ground inputs to drivers for Function Strobe, RDD/WRD, I/O Reset, the 1-Mc Clock, Address Lines 14 and 15, and any unused data lines (including CC3 or CC4 if they are unused).
- Appendix C gives module pin numbers for these cables.
- RDD/WRD is true for WRD.

3.0 LOGIC DESIGN AT THE INTERFACE

Cable drivers may be driven by any standard logic circuit, including flip-flop outputs. However, the cable driver circuit takes eight unit loads. The nominal output of standard SIGMA logic elements is 16 unit loads. Pull-up resistors consume two unit loads, and terminating resistors consume five unit loads. Terminating resistors are not ordinarily needed on circuits that feed cable driver inputs; thus any circuit is apparently allowed to feed one cable driver and six other loads. It is recommended that, where possible, circuits feeding cable drivers not drive any other points.

3.1 Circuitry for Function Strobe Acknowledge

The circuitry for acknowledging the Function Strobe signal is shown in Figure 2-1 on the following page.

3.2 Read Direct Timing

The timing for the Read Direct instruction is shown in Figure 2-2. For this instruction, the following operations take place.

1. The CPU puts the effective address on the Address Lines and holds RDD/WRD at ground for at least 360 nanoseconds, then brings up Function Strobe.

2. The external unit that can properly decode the Address Lines uses Function Strobe as a gate to put the following signals on their proper lines:

   - Data Bits
   - CC3 and CC4
   - Function Strobe Acknowledge

3. After seeing Function Strobe Acknowledge, the CPU delays for a short period (whose duration is not germane to the proper operation of external units) and accepts the data and CC3 and CC4; then it drops Function Strobe, after a delay of 300 nanoseconds minimum (no maximum).

4. When the addressed external unit sees Function Strobe drop, it immediately releases the data lines, CC3, CC4, and Function Strobe Acknowledge.

5. The CPU does not drop the address lines until (200 nanoseconds minimum) it sees Function Strobe Acknowledge released.

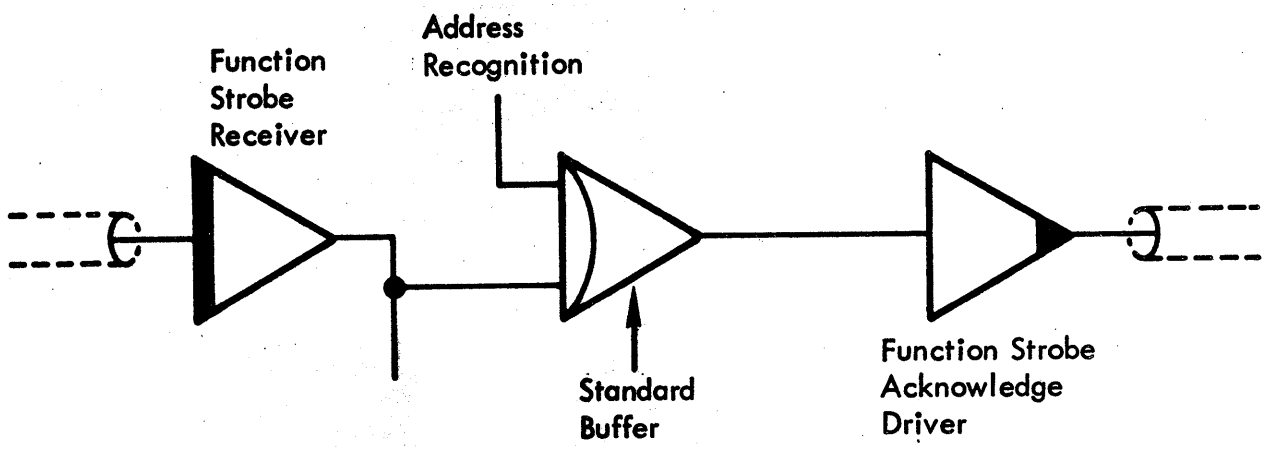


Figure 2-1. Function Strobe Acknowledge Circuit

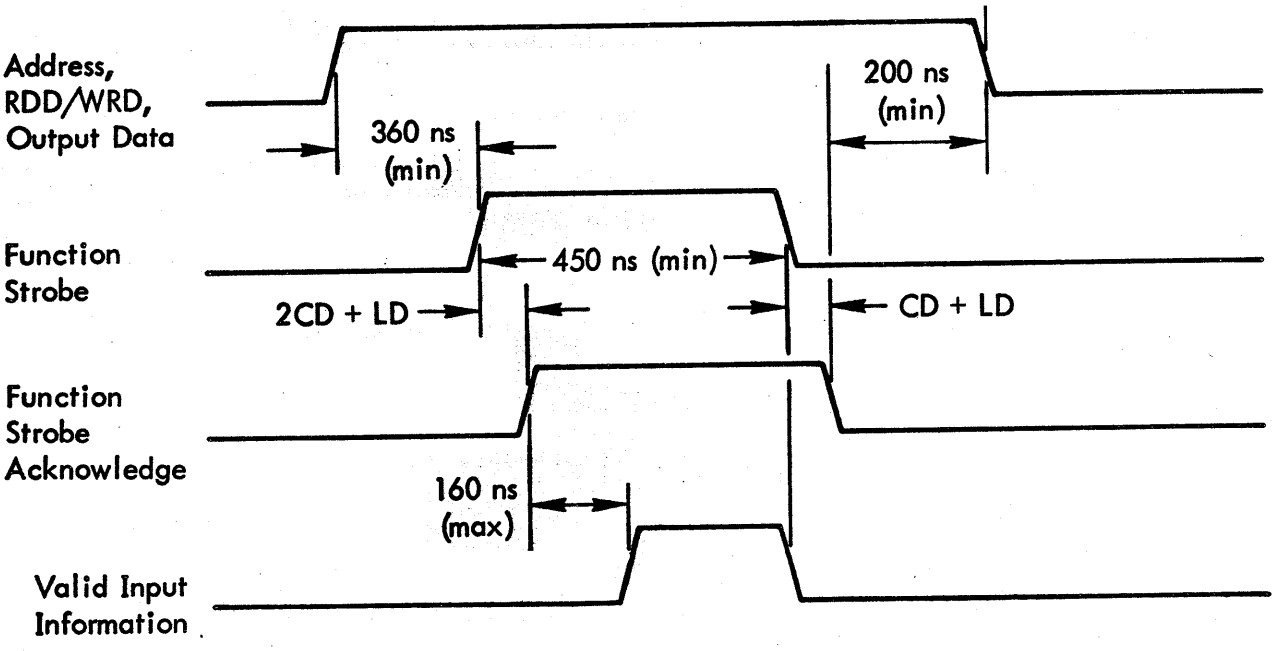


Figure 2-2. Read Direct Timing and Write Direct Timing

3.3 Write Direct Timing

The timing for the Write Direct instruction is shown in Figure 2-2. For this instruction, the following operations take place:

1. The CPU puts the effective address on the Address Lines and holds RDD/WRD at a true state for at least 360 nanoseconds, then brings up Function Strobe.
2. The external unit that can properly decode the Address Lines accepts the information on the data lines and uses Function Strobe as a gate to put CC3, CC4, and Function Strobe Acknowledge on their lines. The output data may be accepted either by using Function Strobe as a dc gate (two-sided loading), or by ac clocking at the trailing edge of Function Strobe. The latter approach is recommended.
3. After seeing Function Strobe Acknowledge, the CPU delays for a short period (whose duration is not germane to the proper operation of the external unit), and accepts CC3 and CC4. Then it drops Function Strobe, after a delay of 300 nanoseconds minimum (no maximum).
4. When the addressed external unit sees Function Strobe drop, it immediately releases CC3, CC4, and Function Strobe Acknowledge.
5. The CPU does not drop the address lines, the data lines, or RDD/WRD, until it sees Function Strobe Acknowledge released (200 nanoseconds minimum).

3.4 Hanging Up the RWD Interface

It is a Systems Engineering policy not to hang up the RWD interface for any duration longer than 1.6 μsec, and only then in cases where there is clearly an engineering or economic advantage to be gained by doing so (for instance, to avoid the use of one-shots or unnecessary buffer storage). The recommended logic for accomplishing this is shown in Figure 2-3 on the following page, along with timing diagrams. Point B is true for 0.6 to 1.6 μsec.

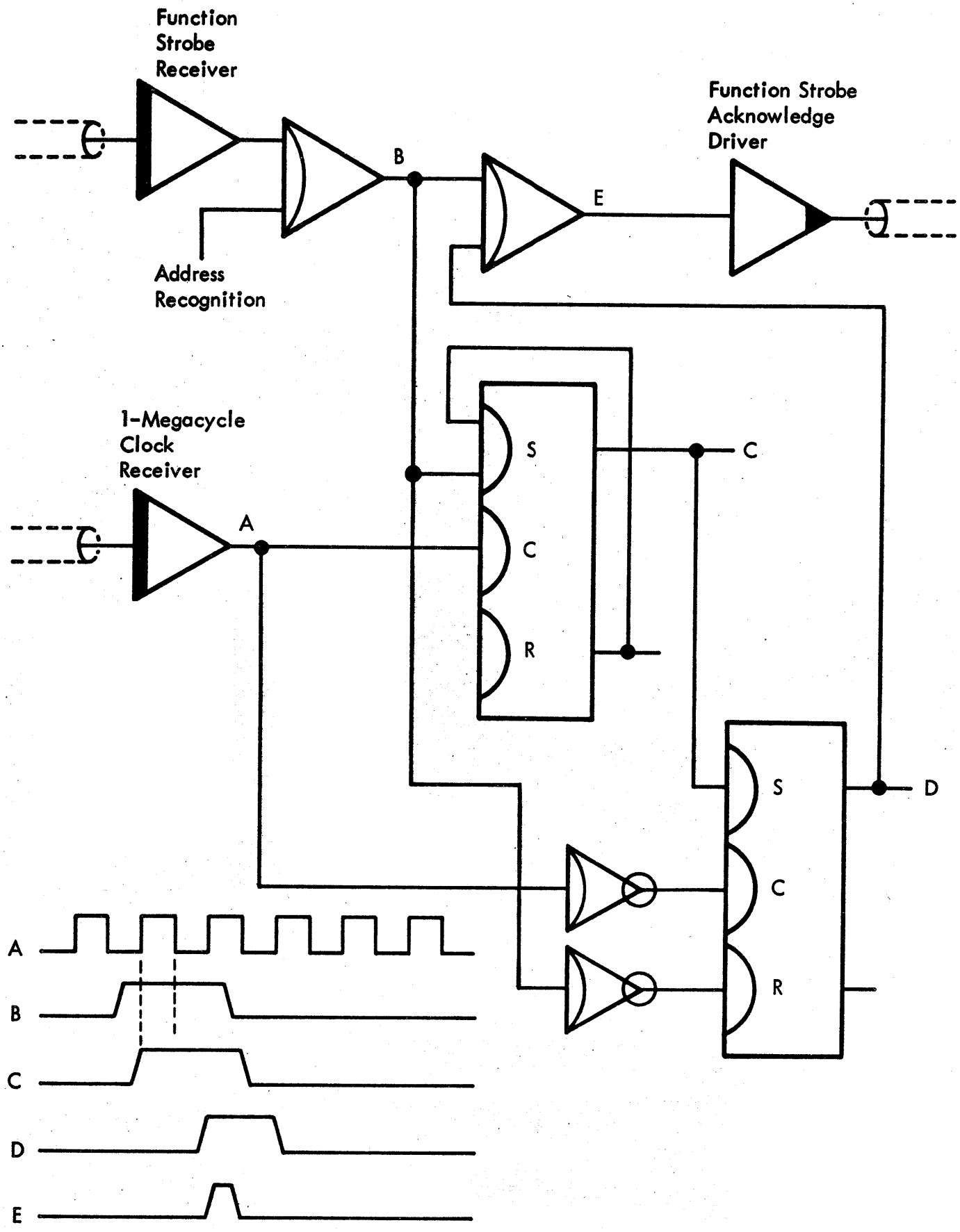


Figure 2-3. RWD Interface Delay Logic

3.5 Decoding

All units designed by Systems Engineering must completely decode all 16 bits of the RDD and WRD effective address. There should be absolutely no exceptions to this rule.

4.0 RECOMMENDED METHOD FOR ACHIEVING SIGMA 2 COMPATIBILITY

Special units designed expressly for SIGMA 2 need not give any consideration to potential usage on SIGMA 5 or SIGMA 7. In the event that they were so used, the only disadvantage would be a reduction in efficiency, since only half of each SIGMA 5 or SIGMA 7 word would be used. There are no engineering barriers to such interchange.

Special units designed expressly for SIGMA 5 or SIGMA 7 should give some consideration to potential usage on SIGMA 2. If such units are optimized for operation on SIGMA 5 and SIGMA 7, they will, in general, not be able to operate effectively on SIGMA 2 unless the usage of each half of each data word is independent of the usage of the other half. The information in the following paragraphs regarding compatibility for standard units should be considered in designing special units for SIGMA 5 and SIGMA 7, since it may be profitable in some cases to make even such special units compatible with SIGMA 2.

All standard units must be capable of efficient operation on either SIGMA 2 or SIGMA 5 and SIGMA 7, unless quite strong economic or operational disadvantages can be shown. The scheme used in the JB30/JB31/JB32 Digital Junction Boxes to ensure this compatibility is described below.

Two bits are reserved in the effective address of each RDD or WRD instruction to specify whether the operation should be only on the high half of the word, only on the low half, or on the full word (both bits true). In general, the mechanics of implementation have shown that there is small incremental cost increase associated with this approach.

A slot is reserved in each chassis, to which all 32 data receiver outputs (the output data from the CPU) are brought. When the unit is used on SIGMA 2, a special jumper module is inserted in this slot. The module connects bits 00 and 16, 01 and 17, etc. Thus any output operation in SIGMA 2 involves the proper 16 bits from the CPU, whether high-half or low-half operation is specified in the instruction.

Figure 2-4 on the following page illustrates how the same operation is accomplished for a typical application – namely digital inputs. The example shows bits 00 and 16; the remaining pairs of bits up to 15 and 31 are similar.

Figure 2-4 shows the normal implementation for SIGMA 5 and SIGMA 7. When the unit is used on SIGMA 2, the circuits designated "A" are replaced by inverting (rather than buffering) circuits with identical gating structures. Most SIGMA logic modules come in pairs like this. The special module designated "B", which has only jumpers in SIGMA 5 and SIGMA 7, is replaced for SIGMA 2 by one that contains inverters, as shown in Figure 2-4. Thus, no wiring changes of any kind are required.

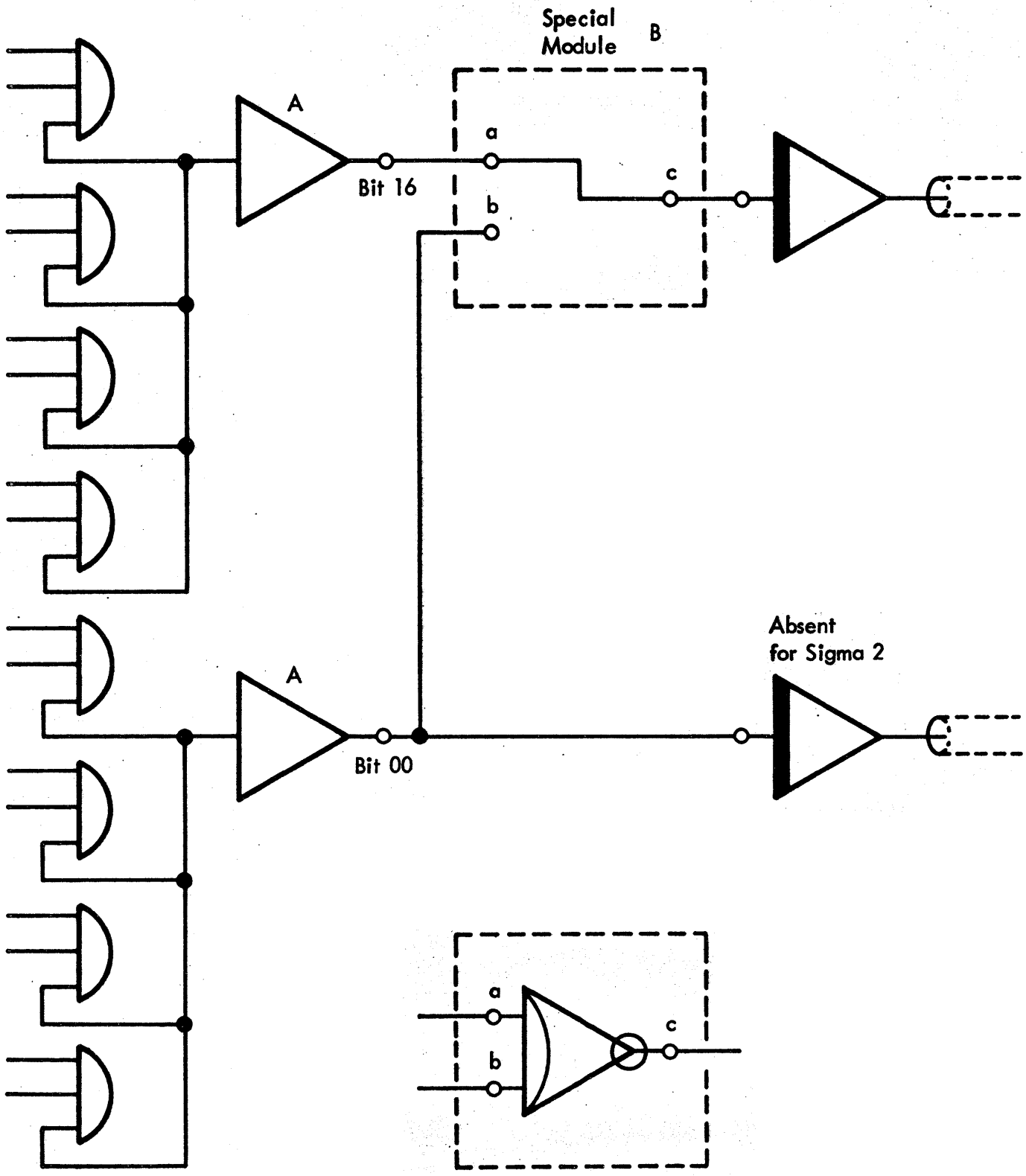


Figure 2-4. SIGMA 2, RWD Compatibility

SECTION III: IOP INTERFACE

1.0 INSTRUCTIONS IN THE COMPUTER

1.1 General Description

SIGMA 5 and SIGMA 7 require different CPU programming than SIGMA 2 to handle the IOP interface. The description that follows refers to SIGMA 5 and SIGMA 7. The SIGMA 2 description is presented in Section III-4.

SIGMA 5 and SIGMA 7 actually communicate only with the IOP. The IOP then handles communication with each Device Controller, including order and control information as well as data transfers. To perform these functions, the IOP utilizes the system's main memory, as well as its own internal fast memory. The internal fast memory in the IOP, which is organized in such a manner that there are a certain number of bits reserved for each possible I/O channel (Device Controller), is not directly accessible to the programmer. However, certain information in this fast memory can be transferred to fast memory or CPU registers by means of the instructions described in this section.

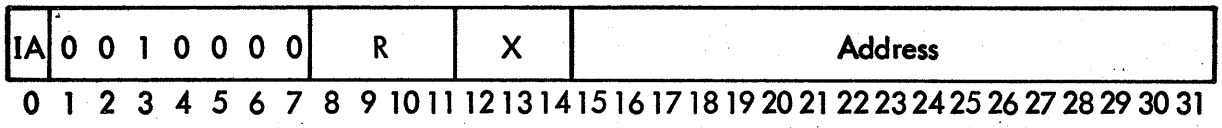
To initiate data transfers with any Device Controller, the programmer may issue a Start I/O (SIO) instruction. This instruction activates the selected Device Controller and provides information regarding how much data is to be transferred and the main memory locations involved in the transfer. It also provides control information that tells the IOP what to do in the event of certain conditions, such as running out of data, various errors, or acting on special control information provided by the Device Controller itself.

The Start I/O instruction does not itself provide any information except an IOP address, a device address, and a memory location that defines the starting point for the actual list of commands stored in main memory. This command list in main memory consists of pairs of words (called command pairs), each of which contains a byte address, a byte count, a command (read, write, read backward, etc.), and various control flags that influence IOP operation. The IOP operates in such a manner that it can read through

this sequence of command words in main memory, executing the functions required by each command pair in turn. The command word format is described in detail in Section III-1.5.

1.2 SIO Instruction

The Start I/O instruction has the following format in a computer program:

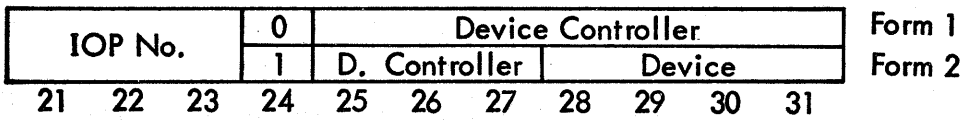


IA = indirect addressing bit

R =, register on currently selected page

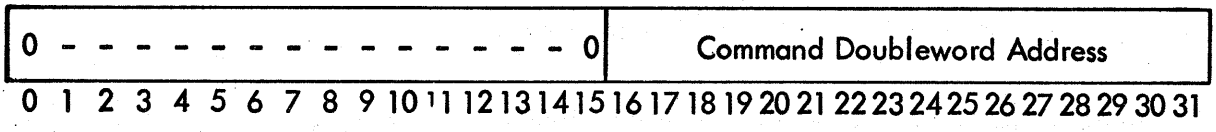
X = index register (none if X = 0)

First, indirect addressing is performed, if IA = 1, to obtain a new address. If indexing is also specified, the contents of the specified index register are added to this new address to form the effective address for the instruction. The low-order eleven bits of this effective address are now interpreted as an I/O address, in the following manner:



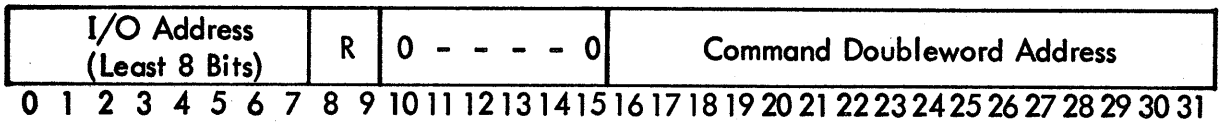
The second form, where bit 24 = 1, is used for Device Controllers that may have many devices attached to them, only one of which can be engaged in data exchanges at any given time. A tape controller and individual tape transports exemplify this type of unit.

SIO assumes that the programmer has previously set up main memory cell 20* with a word having the following format:



*base 16

The SIO instruction now takes the I/O address, as previously determined, and the nature of the R field, and merges them into cell 20 to produce a word in cell 20 having the following format:

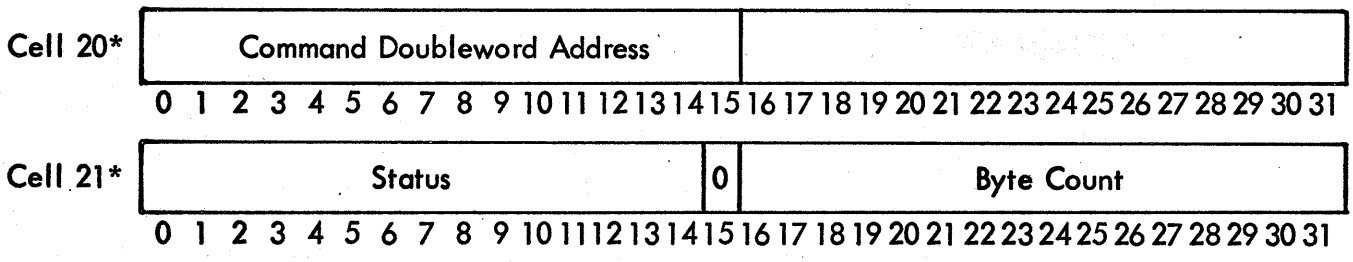


The IOP interprets bits 8 and 9 above in the following manner:

00	The IOP only sets the condition code bits based on the response from the addressed Device Controller (R was 0).
01	The IOP sets the condition code bits as for 00, but also writes status information in main memory cell 21 (R was odd).
10	The IOP sets the condition code bits as for 00, but also writes status information in main memory cells 20 and 21, replacing the original contents above of cell 20 (R was even and non-zero).

The information that the IOP may write back into condition code bits 1 and 2, and cells 20 and 21 (if directed as above to do so), has the following format:

CC1	If set, the I/O address was not recognized by any device.
CC2	If set, the SIO instruction was unsuccessful (i.e., it was rejected by the Device Controller for one of a variety of reasons that are discussed later).



The command doubleword address that is returned to cell 20 is the one associated with the previous order or I/O function to the addressed channel. This is undefined if the instruction is the first SIO since power-on. The byte count returned to cell 21 is the byte count stored in the IOP fast memory word associated with the addressed channel. It is also undefined if this is the first SIO since power-on. The status bits in cell 21, as returned by the IOP (which generates them itself or receives them from the Device Controller), have the following significance:

Bit 0	device interrupt pending
Bits 1 & 2	00 = device ready 01 = device not operational 10 = device unavailable 11 = device busy
Bit 3	device automatic
Bit 4	device unusual end
Bits 5 & 6	00 = Device Controller ready 01 = Device Controller not operational 10 = Device Controller unavailable 11 = Device Controller busy
Bit 7	undefined
Bit 8	incorrect length (via order-in from Device Controller)
Bit 9	transmission data error (data-in, parity, or order-in error)
Bit 10	transmission memory error (data parity)
Bit 11	memory address error (non-implemented memory address)
Bit 12	IOP memory error (memory parity error while fetching command)
Bit 13	IOP control error (two successive Transfer in Channel commands)
Bit 14	IOP halt

Since the IOP might be busy with a Device Controller (exchanging data) when the SIO is issued by the CPU, it is possible for the system to "stall" for a short time until the IOP is free to process the SIO. However, this time cannot, even in the worst case, exceed the time required to finish servicing the currently connected Device Controller, since the CPU then has higher priority (for IOP usage) than any Device Controller that needs service.

After this phase of the SIO instruction is completed, including status exchange with the Device Controller, the CPU copies cell 21 (only if it was written into by the IOP) into register Rul (If R is even, Rul = R + 1; if R is odd, Rul = R). The CPU also copies cell 20 (only if it was written into by the IOP) into register R. If R is odd (making R = Rul), R will then contain the original contents of cell 21, and cell 20 will not be in any of the registers.

1.3 Operational Sequence after SIO

Once a Device Controller has been activated by an SIO instruction, and until it either halts itself or is halted by an HIO instruction, it may make service calls to the IOP. A Device Controller makes a service call by activating a specified line in one of the cables to the IOP, and it may do this at any time, without regard to the state of the IOP. For example, when a Device Controller makes a service call the IOP may be occupied at the time in communicating with the CPU, main memory, or some other Device Controller. A single line is used by all Device Controllers to make their service calls, i.e., they all use the same line.

As soon as the IOP has completed its previous activity, it continues scanning the service call line until it detects activation by some Device Controller. At this time, the IOP does not know which Device Controller is requesting service — any number of Device Controllers might be requesting service simultaneously. The IOP then issues a signal called Function Strobe, in conjunction with a function indicator called Acknowledge Service Call (ASC), which is transmitted to all units attached to the IOP.

A separate, hard-wired priority determination chain runs from controller to controller (this is discussed in detail in Section III-3. 1). This chain is important primarily when more than one controller has called for service. The highest priority controller that has requested service then places its device address on the eight function response lines of the interface, and acknowledges the function strobe. All other units must stay off all lines. The active unit is then connected to the IOP for service, and goes through a service cycle which may involve the transfer of one to five bytes of information.

The IOP uses the address that is returned on the function response lines to select the fast memory location from which required information will be obtained for the subsequent service cycle to the connected controller. The connected controller then initiates data transfers by activating a line designated "Request Strobe". During Request Strobe activation, the Device Controller must also control two lines called Data-Order Request (DOR) and Input/Output Request (IOR). These two lines are decoded by the IOP to select one of four possible functions that may be performed by the IOP during that cycle:

DOR	IOR	Function
0	0	Data Input
0	1	Data Output
1	0	Order Input
1	1	Order Output

The operation of the IOP while performing these functions is discussed later in this section. The manner in which a service cycle is terminated is controlled by two additional signals: End Data (ED) and End Service (ES), coded as follows:

ED	ES	Meaning
0	0	The controller must generate another Request Strobe and accept or provide another byte of data.
1	1	This is the last data byte; the controller must disconnect itself and not generate any more Request Strobes.
1	0	This is the last data byte; however, the controller must not disconnect itself yet and must generate one more Request Strobe. During the next Request Strobe, the IOP will output certain control information to the controller. This is known as a Terminal Order.
X	1	This configuration defines the Terminal Order when preceded by "10"; the controller must disconnect itself and not generate any more Request Strobes.

In response to each Request Strobe generated by the Device Controller, the IOP generates a signal called Request Strobe Acknowledge. Along with Request Strobe Acknowledge, the IOP either provides data (if the subcycle is for Data Out, Order Out, or Terminal Order) or accepts the data provided by the controller (if the subcycle is for Data In or Order In).

The ED line may be driven by either the IOP or the Device Controller. The Device Controller normally drives this line, at the same time it activates Request Strobe, when it has received all the data necessary (or input all the data called for during the current service cycle). It may also drive this line because an error has been detected in data transmission and the Device Controller therefore wishes to terminate operations.

The IOP may drive the ED line true for one of three reasons:

1. An error of some sort has been detected, and the IOP has been conditioned to terminate operations when this error is detected.

2. A word boundary has been reached during data transfers, and the IOP would have to go back to main memory to get more data bytes. Activating ED and ending the service cycle permits other operations, such as service to the CPU or other Device Controllers, to take place.

3. The service cycle was for either Order Out or Order In; each consists of a one-byte transfer.

The ES line, on the other hand, may be driven only by an IOP. It is driven under precisely two circumstances:

1. Either the IOP or the Device Controller has signalled ED, and there is no necessity for a Terminal Order.

2. It is driven during the Terminal Order.

In the first case, ES is driven during the same subcycle that ED was driven, so they both appear together back at the Device Controller. The Device Controller should use the trailing edge of Request Strobe to inspect the state of ED and ES and decide whether or not additional Request Strobes will be required and if one is required, whether it should be for a Terminal Order or for more data transfer.

In SIGMA 5 and SIGMA 7, Order In is always followed by a Terminal Order, even though there may sometimes be no important information to be transmitted in the Terminal Order.

It should be noted that in SIGMA 5 and SIGMA 7, precisely seven different types of service cycles can occur:

1. one, two, three, or four data bytes out

2. one, two, three, or four data bytes out and a Terminal Order

3. one, two, three, or four data bytes in

4. one, two, three, or four data bytes in and a Terminal Order

5. Order Out (one byte)

6. Order In (one byte) and a Terminal Order

7. Order Out and a Terminal Order

The following points should be noted carefully:

- The coding of DOR and IOR by the Device Controller need only be given during the first byte of each service cycle, even in the cycle is five bytes long; the state of these lines is ignored after the first byte of each service call.

- The Device Controller, if it wishes to end the service cycle, drives ED while the last data byte is being input or output; the Device Controller should not drive ED during the Terminal Order.

- The Device Controller must disconnect following the subcycle in which it sensed ES true, and must generate an additional Request Strobe following any subcycle in which it sensed ES false.

1.4 Data and Command Chaining

"Data chaining" is the term used to describe a particular IOP reaction to "running out of data", i.e., the byte count in IOP fast memory for a particular channel goes to zero. If the programmer has specified data chaining with bit 0 of the odd command word, then the IOP performs the following sequence of actions when the byte count goes to zero:

1. The IOP fast memory is accessed to determine the last command doubleword address.

2. This address is incremented by two, to produce a new command doubleword address.

3. The IOP accesses two main memory cells (the command doubleword) and stores in its own fast memory a new byte count, a new memory byte address, and a

new set of eight IOP control flags. The "order" bits in the command doubleword are ignored, and not passed on to the Device Controller.

4. The IOP then continues data exchanges with the Device Controller, based on further service requests from that Device Controller. The Device Controller has no knowledge of the occurrence of data chaining.

"Command chaining" is caused by the Device Controller, in response to orders from the programmer. Its net effect is similar to that of data chaining, except that the IOP also passes along a new "order" (the highest eight bits of the first command word) to the Device Controller. Command chaining only occurs as a result of the following sequence of operations:

1. The Device Controller signals "channel end" by means of an Order In.
2. If command chaining was called for by bit 2 of the second command word of the last executed command doubleword, the IOP sets the command chaining bit in the Terminal Order that follows Order In.
3. The Device Controller recognizes this condition and requests an Order Out cycle. At this time the IOP determines the new doubleword address and accesses main memory to get a new command doubleword, just as it did in the data chaining operations described above.

Data chaining is used for scatter-read or gather-write operations, where the external device is operating with continuous data that may come from or be delivered to non-contiguous areas of memory, in sub-blocks of any size specified by the programmer. Command chaining provides a means of writing a sub-program (a sort of subroutine) whose execution and sequencing can be under control of the IOP and/or the Device Controller. It avoids the necessity for active program intervention in many cases, thus saving time and programming effort.

The IOP does not ordinarily take any action based on the particular order in the command doubleword, except for a Transfer in Channel command. This command is not passed on to the Device Controller. Instead, the IOP alters the current command doubleword

address to the new location specified by the Transfer in Channel (TIC) command. When TIC is used in conjunction with the chaining modifier bit (which the Device Controller can signal in an Order In), it allows selective branching within the CPU command list, under control of the Device Controller.

1.5 Command Word Format

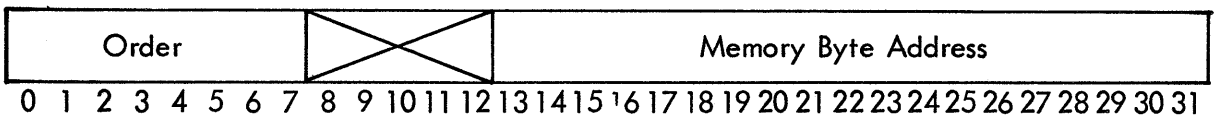
The IOP accesses main memory to obtain command words under two conditions:

- The Device Controller has made a request for Order Out.
- During data exchange (input or output), the IOP determines that the byte count has gone to zero, and the data chaining bit in the previous command word was set.

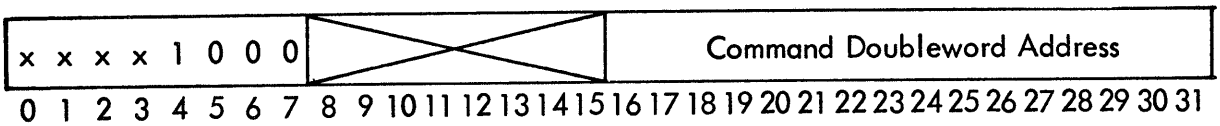
The request for Order Out in the first case above can be generated by the Device Controller for one of two reasons:

- An SIO instruction was issued to and accepted by the Device Controller. The Order Out is thus issued to obtain the first command doubleword from main memory, in order to inform the Device Controller regarding the exact nature of the operations it is to perform and to allow the IOP to store the required control information (byte count, memory byte address, control flags).
- The Device Controller had issued an Order In specifying "channel end", and in the Terminal Order following that Order In, the IOP had indicated that the programmer had called for command chaining.

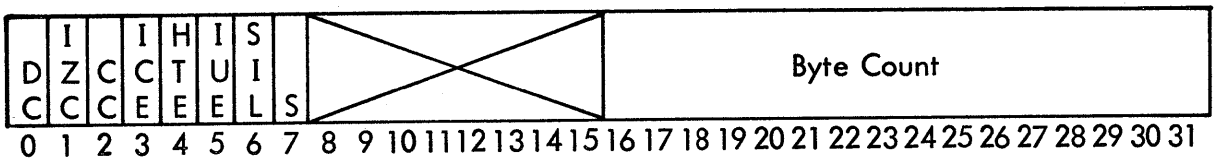
The general format for command words is as follows:



For Transfer in Channel commands, the format is:



Odd command words have the following format:



The coding of the Order bits in the even command word is:

MMMMMM11 = Control

MMMMMM01 = Write

MMMMMM10 = Read

MMMM1100 = Read Backward

MMMM0100 = Sense

XXXX1000 = Transfer in Channel

I0000000 = Stop

where

M denotes a modifier bit unique to the particular Device Controller

X denotes "don't care"

I denotes interrupt condition

The coding of the IOP flag bits in the odd command word is shown in Table 3-1 on the following page.

Table 3-1. IOP Flag Bit Coding

Bit	Name	Function
0	DC	Data Chain: if this bit is true, then when the byte count goes to zero the IOP accesses memory for the next command pair but does not pass the Order along to the Device Controller.
1	IZC	Interrupt on Zero Count: if this bit is true, then when the byte count goes to zero the IOP generates a Terminal Order to the Device Controller, in which the interrupt bit (data line 0) is set.
2	CC	Command Chain: if this bit is true, then when the Device Controller signals "channel end" to the IOP (by setting data line 3 in an Order In), the IOP signals command chaining to the Device Controller in the Terminal Order which follows the Order In (by setting data line 2 in that Terminal Qrder).
3	ICE	Interrupt at Channel End: if this bit is true, then when "channel end" is signalled to the IOP by the Device Controller (by setting line 3 in an Order In), the IOP sets the interrupt bit in the Terminal Order which follows that Order In.
4	HTE	Halt on Transmission Error: if this bit is true, then the IOP signals "IOP halt" to the Device Controller by setting bit 3 in a Terminal Order, if any of the following conditions occur: . A parity error is detected by the IOP during communication between the IOP and main memory. . The Device Controller signals Transmission Error to the IOP by means of bit 0 set in an Order In. . The Device Controller signals Incorrect Length to the IOP by means of bit 1 set in an Order In and the SIL flag is not set.
5	IUE	Interrupt at Unusual End: if this bit is true, then when Unusual End is signalled to the IOP by the Device Controller in an Order In, the IOP sets the interrupt bit in the Terminal Order that follows the Order In.
6	SIL	Suppress Incorrect Length: if this bit is true, then Incorrect Length indications from the Device Controller during Order In are ignored. Otherwise, Incorrect Length indications may cause an IOP Halt to be issued to the Device Controller if the HTE flag is set (see above).
7	S	Skip: if this bit is true, the IOP performs all actions usually associated with the command except memory accesses.

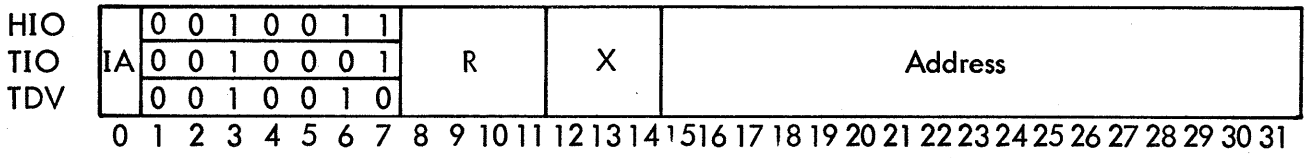
It should be noted that, for each channel (Device Controller), the IOP must store and maintain in its own fast memory the following information:

- Memory Byte Address (19 bits), which is incremented or decremented after each data exchange with the Device Controller (decremented only when Order was Read Backward).
- Command Doubleword Address (16 bits), which is incremented by two prior to the extraction of each new command doubleword from main memory. It is ordinarily changed only by an SIO instruction or this incrementing, but a TIC command may also change it.
- Byte Count (16 bits), which is incremented by one after the exchange of each 8-bit byte with the Device Controller.
- IOP Control Flags (8 bits), which may be altered only by a new command word other than TIC or Stop.

It should also be noted that Order does not have to be preserved in IOP fast memory. The Order is transmitted directly to the Device Controller, if required, at the time the command word is read from main memory by the IOP, and is not needed thereafter.

1.6 HIO, TIO, and TDV Instructions

The format for Halt I/O (HIO), Test I/O (TIO), and Test Device (TDV) instructions is similar to that for SIO.



The low-order eleven bits of the effective address, after indirect addressing and indexing (if specified), are also interpreted (as for SIO) as specifying an IOP number and an 8-bit device address. Memory cell 20* does not have to contain a command doubleword address, as for SIO.

*base 16

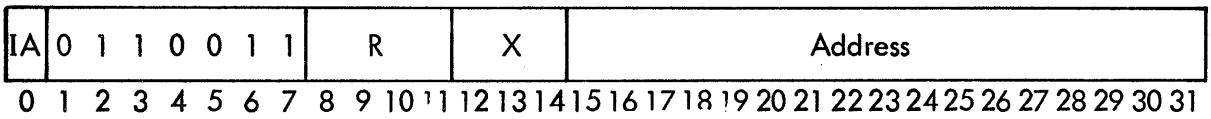
However, the R field is still used to specify the locations to which the IOP returns data (20 only, 20 and 21, or neither). The format of cell 20 before the execution of HIO, TIO, or TDV is the same as for SIO (device address and R bits), without the command doubleword address.

The contents of cells 20 and 21 after HIO, TIO, or TDV has been executed are similar to the format after SIO, described in Section II-1.2. The status bits (0-14 of cell 21) have exactly the same meaning for SIO, HIO, and TDV. For TDV, the high-order eight status bits represent status information unique to the particular Device Controller that is being tested. Descriptions of this status are contained in the individual specifications for each controller.

1.7 Interrupt Structure and AIO Instruction

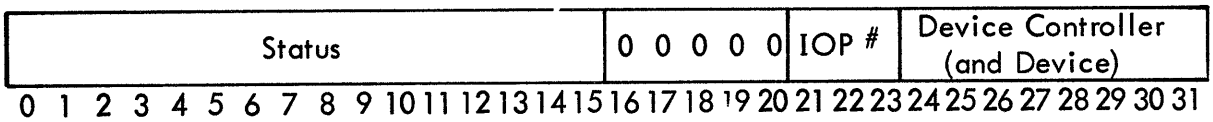
The IOP has its own priority interrupt structure which is completely independent of the normal external priority interrupt structure, except that the relative priority of the two groups may be altered by hard-wiring. All interrupts from Device Controllers attached to IOP's come through a single memory cell. This is equivalent to a situation where all peripheral equipment interrupts (including IOP interrupts such as Count Equals Zero) are bussed to a common priority interrupt, whose place in the priority chain is variable (hard-wired only). There must, therefore, be some way to determine which of many possible devices has actually caused the common interrupt; in fact, there may be at any given time active interrupt requests from many controllers, but only one interrupt occurs.

The Acknowledge Input/Output Interrupt (AIO) instruction is used to determine the source of such an interrupt request. This instruction is not addressed to a particular device, since one of its results is to determine which device caused an interrupt. Thus, the instruction format is somewhat different from SIO et al:



The difference from SIO is primarily in the interpretation of the effective address: only bits 21 to 23 are significant. These bits establish the type of interrupt being acknowledged. For standard I/O interrupts, these bits should be coded 000. The remainder of the address field is not used.

When an AIO instruction is issued, various control signals are sent out to the IOP's (up to eight in a system). Each IOP passes the control signals on to the next IOP, unless one of the controllers connected to it has an active interrupt. The IOP with the interrupt-requesting controller then passes the control signals on to its Device Controllers, where the controllers themselves establish priority in a similar manner. The highest priority interrupt-requesting Device Controller sends its own address to the IOP, along with some status information regarding its interrupt request. The IOP appends its own IOP number, and passes this information to the CPU, which writes into cell 20 a word of the following form:



Status bits 0-7 represent interrupt status information unique to the Device Controller whose interrupt was acknowledged. The significance of these bits is described in the specifications for each device. Bits 8-15 have standard meaning for all IOP's and controllers:

- Bit 8 Incorrect Length, as defined for SIO
- Bit 9 Transmission Data Error, as defined for SIO
- Bit 10 Zero Byte Count Interrupt (set only if the IZC flag in the last command doubleword was true and the byte count equals zero)
- Bit 11 Channel End Interrupt (set only if the ICE flag in the last command doubleword was true and the device has signalled "channel end" to the IOP via an Order In)
- Bit 12 Unusual End Interrupt (set only if the IUE flag in the last command doubleword was true, and the IOP has signalled the controller to halt by means of a Terminal Order)

This word is then moved to the register specified in the AIO instruction.

There are two basic reasons for which a Device Controller may issue an interrupt request:

- In a Terminal Order from the IOP, the interrupt bit (bit 0 in the Terminal Order) was true.
- An internal condition in the controller calls for an interrupt, based on the design specifications for the Device Controller. In some cases the Device Controller might need or accept control information from the IOP (and hence the CPU and programmer) which conditions this interrupt, or it might be generated internally, by such an event as the depression of a manual push-button.

2.0 INTERFACE SIGNAL INTERACTION AND TIMING

2.1 Priority Determination

Three cables run from the IOP to the first Device Controller, from the first to the second Device Controller, etc., as described in Section I-3. In addition, a priority determination cable runs only from Device Controller to Device Controller, and does not connect to the IOP. The signals on this cable are used to determine which Device Controller reacts to stimuli for two different sets of conditions:

- When more than one Device Controller has made a service request at nearly the same time, only one may be connected for service. When the IOP acknowledges the service call, the highest priority requesting Device Controller responds to the IOP's acknowledgement. The other waits until some subsequent service call acknowledgement comes from the IOP.
- When more than one Device Controller has made an interrupt request, and the AIO instruction is executed by the CPU and IOP, only the highest priority Device Controller may respond to this function; the others must wait until the CPU executes some subsequent AIO instruction.

It should be noted that the physical routing of the priority determination cables, which determine the relative priority of units in the event of simultaneous interrupts or service calls, is completely independent of the physical routing of the three ordinary data transmission cables. These three cables are normally installed according to the physical placement of Device Controllers, in order to minimize total cable length. Maximum allowable cable length is 100 feet. The routing of the priority determination cable is based on the desired sequence of responses in the event of simultaneous requests, and 160 feet is the maximum allowed. The desired sequence of responses is influenced by the likelihood of rate overruns, the frequency of service requests, and the degree of difficulty involved in correcting rate overruns. One proposed approach is that devices which require manual intervention to correct rate overruns (such as punched card devices) should have highest priority for service; it should be noted that rate overruns on magnetic tape and discs can be corrected entirely under program control.

Five signals are carried on the priority determination cable:

- HPI — High Priority Interrupt
- HPS — High Priority Service
- BSY — Busy
- AVI — Available Input
- AVO — Available Output

The first three of these signals each constitute a bus which is tapped and driven by each unit, just like all signals on the ordinary data transmission cables. AVI and AVO are unique in that they are not busses; AVO is an output signal from each Device Controller, sent to the next one down the line; it is a logical function of AVI (an input signal), BSY, HPI, HPS, and the internal state and design specifications of the Device Controller.

When the IOP detects a service request (which may be from one or any number of Device Controllers), it activates a strobe that is fed to all Device Controllers in parallel and an indicator that defines the Acknowledge Service Call function. Only the highest priority device must respond actively, while lower priority devices respond passively. Section III-2. 2 describes this response in more detail.

AVI is always true for the first Device Controller in the priority determination chain. AVI for each subsequent Device Controller is equal to AVO from the preceding Device Controller in the chain. Each Device Controller must look at the required lines and at its own internal signals and produce the following reactions:

1. If a Device Controller has no service request pending, it must pass on AVO as soon as it sees AVI.

2. If a Device Controller has a service request pending, it passes on AVO when it sees AVI only if the following conditions are met:

   - HPS is true, and
   - that unit's service request is low priority (i.e. the unit has not generated HPS itself.

3. If a Device Controller has a service request pending, it does not pass on AVO when it sees AVI if either of the following conditions are true:

   - it has generated HPS itself, or
   - HPS is not true.

   When it does not pass on AVO, it must generate BSY instead.

4. Every unit must perform the required passive acknowledgement (see Section III-2.2) for each of the mutually exclusive possibilities:

   - BSY is seen to go true
   - AVI is passed on as AVO
   - BSY is generated by the unit

5. The unit which causes BSY to go true is then connected to the IOP for service, and may engage in data exchanges; all other units must await their turn at some subsequent time.

The same priority chain is used in connection with response to the AIO instruction, except that a different function indicator occurs (AIO instead of ASC), and HPI is used in place of HPS in the logic. Figure 3-1 on the following page shows schematically the logic implementation for a typical unit. However, the designer will not generally have to implement this, since it is included as a part of the Subcontroller (see Section III-3).

2.2 Leading and Trailing Acknowledgement

Device Controllers in a SIGMA system time-share a single bus for most control and data signals. Obviously, the cable length of this bus is a variable from system to system. To permit completely asynchronous operation on this bus, and therefore to obtain maximum operating rates (IOP bandwidth) for each system, a scheme for positive acknowledgement of both the leading and trailing edges of IOP-generated strobe pulses has been devised.

The IOP issues a Function Strobe during the execution of each of the five computer instructions, and also for the Acknowledge Service Call function. This strobe must be acknowledged by every device, even though only the device that recognizes its own address is involved in active communication with the IOP, in order to allow execution of the instruction. The IOP observes the acknowledgement, and uses this to drop the Function Strobe. Each device must indicate that it has seen the release of Function Strobe (and is therefore finished with its required reactions) before the IOP completes and exits the instruction.

Due to delays in signal transmission over long cables, devices that are physically closest to the IOP sense state changes in Function Strobe (and other IOP signals) sooner than those devices that are farther away. This would ordinarily cause IOP counter-reactions before the Function Strobe signal had propagated to the end of a long cable. Hence, the IOP could possibly enter the execution phase of an instruction before the farthest device in a system had even seen Function Strobe; this would obviously be improper, and could lead to errors.

To avoid this situation, a scheme has been devised to notify the IOP of receipt of both the leading and trailing edges of Function Strobe by the last (farthest) unit on a line.

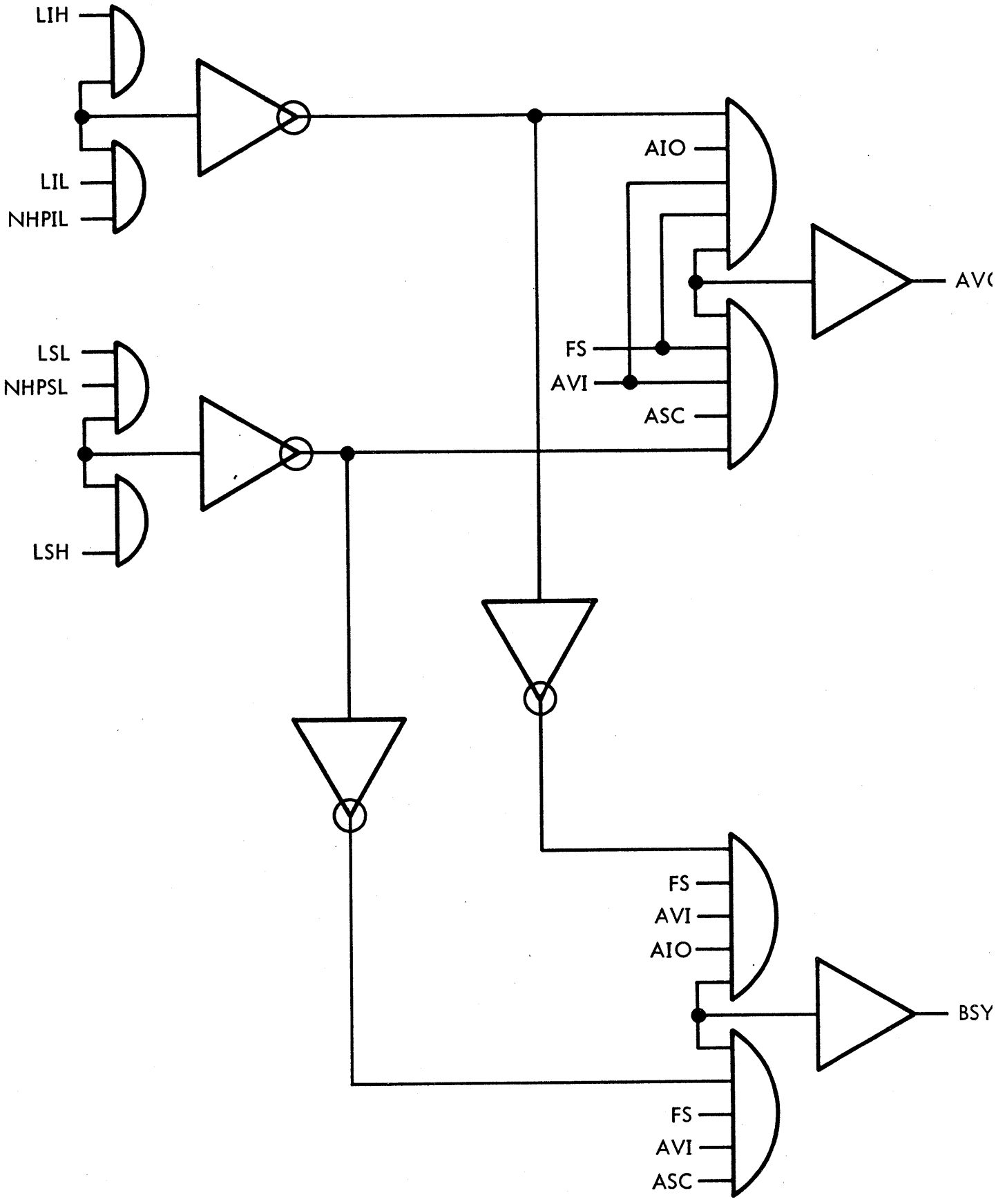


Figure 3-1. Priority Chain Implementation

Two signal lines are required, designated Function Strobe Acknowledge Leading (FSL) and Function Strobe Acknowledge Trailing (FST). Each device, when it senses Function Strobe (FS), must release FSL, which is normally true, and drive FST (normally false). The IOP senses FST to be true as soon as the closest device has driven it, based on the cable-driving scheme described in Section I-3. However, the IOP cannot sense FSL false until all devices (hence the last one, which is presumably the farthest) have released it. Thus the IOP correctly assumes that the true-to-false transition of FSL indicates that all devices have seen FS.

When the IOP has released FS (after it has seen FSL go true), each device then drives FSL again, and releases FST. As soon as the first device drives FSL, the IOP senses it. However, that is valueless for determining when the last device on the cable has seen FS go low. On the other hand, FST cannot go low until every device, including the last, has released it, since any one driver true would hold the line true. Thus the IOP correctly assumes that the true-to-false transition of FST indicates that all devices have properly sensed the false state of FS.

Figure 3-2 shows a timing diagram for the interaction of FS, FST, and FSL. It should be noted that if there is not a device present with the specified address, neither the IOP nor the CPU hangs up, since all devices are controlling FST and FSL (whether or not they are addressed) to allow this operational sequence to take place. One of the condition code lines is also driven only by the device which recognizes its own address. Thus, the absence of a device is signalled by the failure of any device to drive this condition code line.

2.3 Processing Computer Instructions

The four instructions SIO, HIO, TIO, and TDV are quite similar. Their primary characteristics and differences are listed below:

- SIO causes the addressed Device Controller to be started (although the SIO may be rejected for certain reasons). SIO also provides the first command doubleword address to be stored in IOP fast memory.

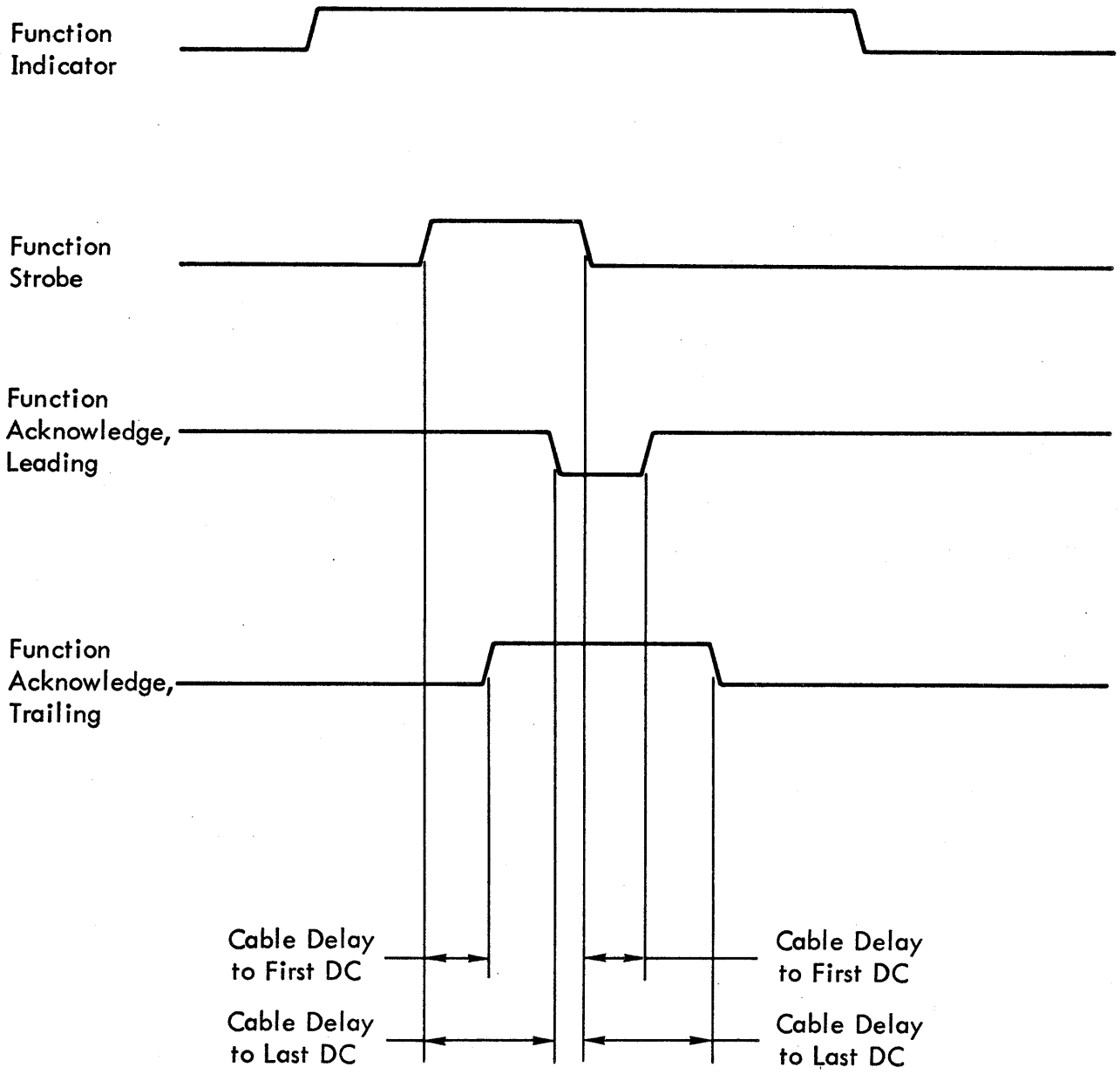


Figure 3-2. Function Strobe Acknowledgment

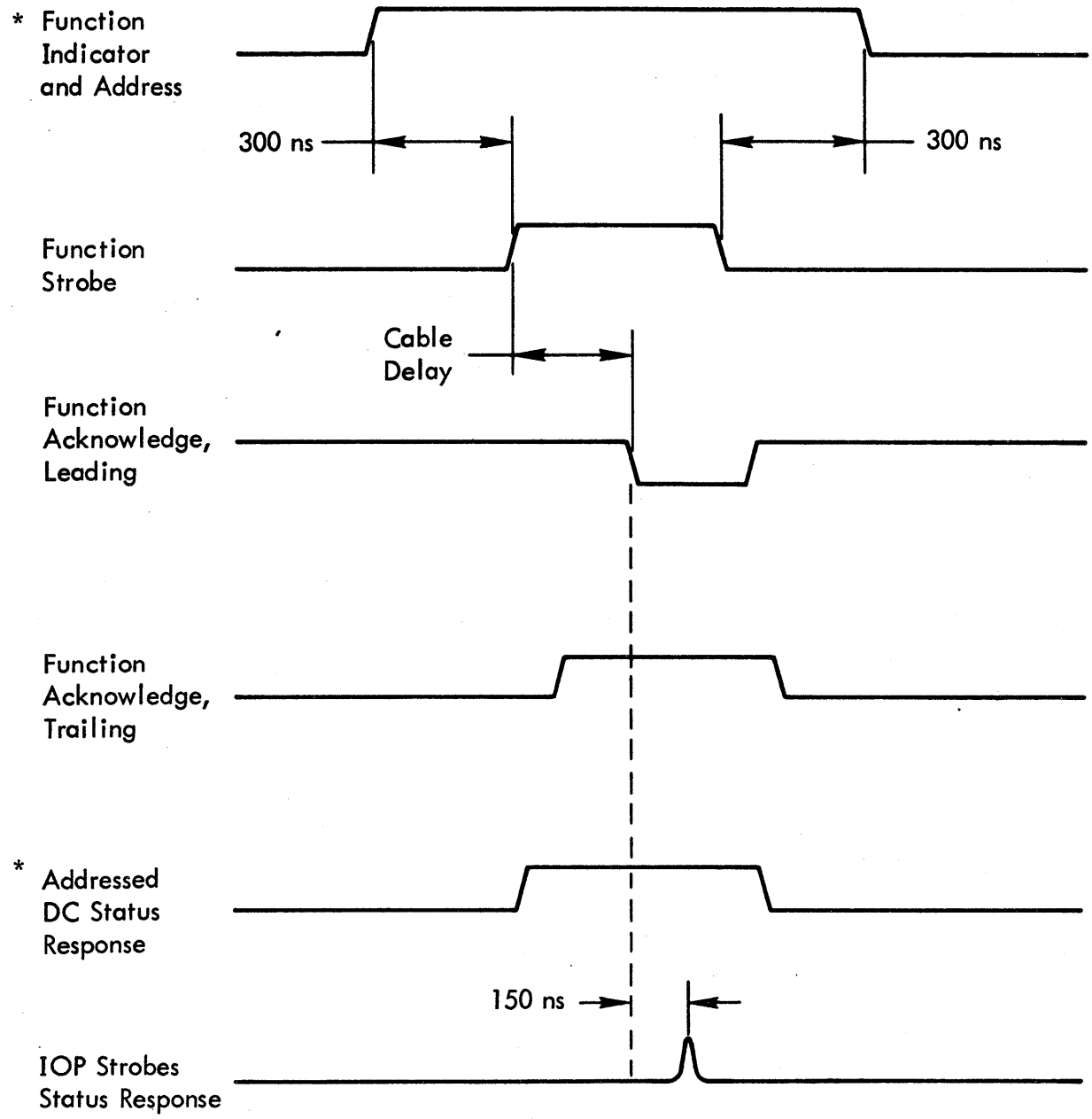
- HIO causes an unconditional halt of the Device Controller and the device.
- TIO and TDV differ only in that the status information returned to the CPU during TIO is general and independent of the particular Device Controller or device, while the status information returned during TDV is specific for each device (defined in the individual specifications for each Device Controller).

All four instructions are interpreted in exactly the same manner by the CPU, except as described above. IA and X fields are normal, while the R field specifies the extent of status information to be returned to the CPU by the IOP (either none, one, or two words). The low-order eleven bits of the effective address specify an IOP, a Device Controller, and (perhaps) a device.

The four instructions cause the following signal interaction on the IOP bus:

- One of four function indicator lines is driven, to select the particular instruction being executed, and a device address is placed on the data lines.
- The IOP delays 300 nanoseconds, then signals Function Strobe to the Device Controllers.
- All non-addressed Device Controllers must respond by releasing FSL, driving FST, and staying off all other lines.
- The addressed Device Controller must place status information on the function response lines FR0 through FR7 and IOR and DOR, and must release FSL and drive FST.
- The IOP responds to the released FSL by releasing FS.
- All Device Controllers must respond to the released FS by driving FSL and releasing FST.
- The information returned on the FR lines by the addressed Device Controller is defined in Table 3-1 for SIO, HIO, and TIO instructions.

Figure 3-3 shows a timing diagram for a typical SIO instruction execution.



*No output address for AIO or ASC; DC returns its address with status response.

Figure 3-3. Instruction Execution Timing
(SIO, HIO, TIO, TDV, AIO, ASC)

Table 3-1. SIO, HIO, and TIO Status Response

FR0		Interrupt Pending
FR1	00	Device Ready
	01	Device Not Operational
FR2	10	Device Unavailable
	11	Device Busy
FR3		Device Auto
FR4		Device Unusual End
FR5	00	Device Controller Ready
	01	Device Controller Not Operational
FR6	10	Device Controller Unavailable
	11	Device Controller Busy
FR7		Not Assigned

2.4 Service Calls and Acknowledgement

When the IOP senses that the Service Call line is high (denoting that one or more Device Controllers want service), it causes the following sequence of signal interaction on the IOP bus:

- The function indicator line, ASC, is driven true.
- The IOP delays 300 nanoseconds, then signals FS to all Device Controllers.
- All Device Controllers must release the FSL line and drive the FST line true.
- The single Device Controller that is to be serviced (based on examination of the priority determination signals) must put its own address on the FR lines.
- The IOP responds to the released FSL by releasing FS.
- All Device Controllers respond by releasing FST and driving FSL.
- The Device Controller which is to be serviced connects itself to the IOP, and may begin issuing request strobes, as described in Section III-2.5.

It should be carefully noted that all Device Controllers must stay off all lines not mentioned specifically in the foregoing sequence. This allows the ASC function to overlap the previous service cycle to some other connected device, thus increasing IOP bandwidth.

Figure 3-4 shows a timing diagram for a typical ASC function. Section III-2.5, which describes the signal interaction during the service cycle, shows a typical complete cycle (ASC plus the service cycle).

There are cases when it might be desirable to design a Device Controller that would make service calls "all the time" (except when it was in its own service cycle). Such a Device Controller should obviously have the lowest priority in the determination chain; otherwise no other Device Controller could use the IOP simultaneously. However, consideration of the permissible length for the priority determination cable might sometimes make this type of operation inadvisable. Another means is available to allow such a Device Controller to operate and still permit simultaneous action by other Device Controllers. This can be done if the "continuous" Device Controller is inhibited from issuing a service call while any other device has a service call pending. Once the service call has been issued, however, it should be held up until it has been answered and service has been received.

This scheme would be used by a Device Controller that needed to communicate with the IOP as often and as rapidly as possible, without usurping IOP time required by other standard peripheral devices. An example of such a device might be a computer-to-computer coupler.

The High Priority Service (HPS) signal is also available for use by Device Controllers. If used, it must be driven in addition to Service Call (SC). Its effect is only on the priority determination chain, which assigns priority first to all devices that have set HPS, and gives service to a device with the SC set only when no other devices have HPS set.

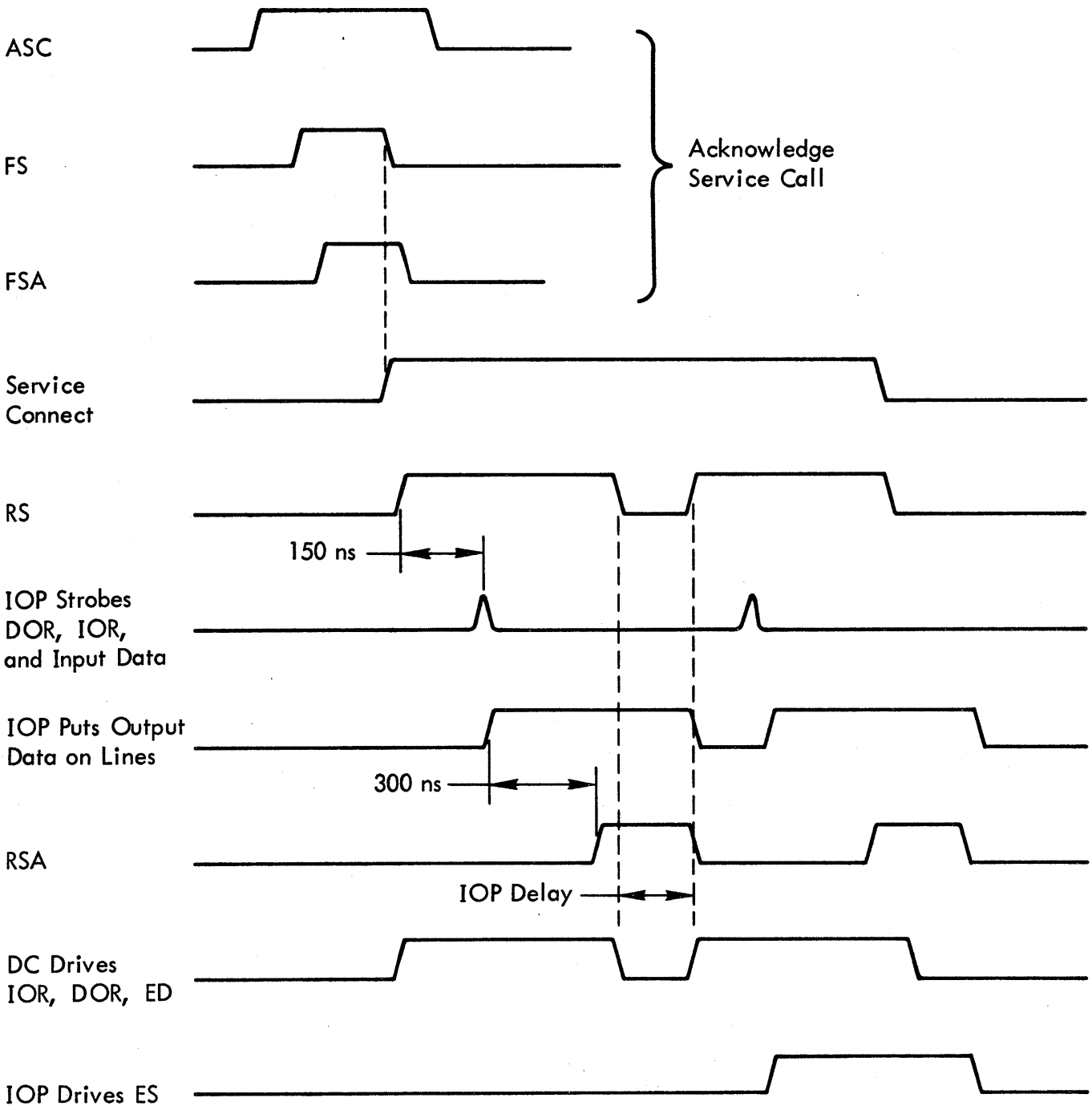


Figure 3-4. Service Cycle Timing

2.5 Request Strobes and Acknowledgement (Data Input and Output)

Once a Device Controller has been connected (i.e., has legally connected itself) to the IOP for service, it may issue Request Strobe (RS) signals. There are 16 lines involved in the IOP/Device Controller communication during a service cycle. These are:

RS	Request Strobe
RSA	Request Strobe Acknowledgement
DOR	Data/Order Request
IOR	Input/Output Request
ED	End Data
ES	End Service
DA0-7	Data Lines
DAP	Data Parity
PC	Parity Check

The sequence of interaction between the IOP and a Device Controller during a service cycle is as follows:

- The Device Controller brings up RS, and at the same time controls DOR and IOR to specify the type of information to be exchanged:

DOR	IOR	
0	0	Data Input
0	1	Data Output
1	0	Order Input
1	1	Order Output

- If an input operation is specified (IOR = 0), the Device Controller at the same time put data (and perhaps data parity) on the data lines, but may drive PC if no parity checking by the IOP is needed.

- The IOP responds to RS by generating RSA. If an output operation was specified, the output data is first put on the lines (after a 150-nanosecond delay), followed by RSA after an additional 300-nanosecond delay.

- When the Device Controller sees RSA, it must release RS. If an output operation was specified, this is the point at which output data should be strobed.

- When the IOP sees RS released, it releases RSA. The Device Controller may then generate another RS, subject to the state of ED and ES, as described below.

- When the Device Controller generates RS, it may also drive ED if it needs the current data exchange to be the last one. However, the Device Controller is allowed to disconnect itself only if it sees ES true.

- Therefore, under ordinary circumstances, it the IOP sees ED true, it drives ES true (with the same timing as output data), notifying the Device Controller that disconnection is required. The IOP may also drive ED true itself (with or without ES) to terminate the data exchange portion of the service cycle.

- Under certain circumstances, the IOP may have control information it needs to transmit to the Device Controller (other than Order Output). This is accomplished by means of a Terminal Order. Whenever the Device Controller sees ED true and ES false, it must remain connected and generate exactly one more RS, in order to receive the Terminal Order.

- The timing for this additional Terminal Order subcycle is identical to all others; the Terminal Order data timing is just like Order Out or Data Out. During the Terminal Order, ES is caused to go true by the IOP; the Device Controller must disconnect itself after a Terminal Order.

Figure 3-4 shows a timing diagram for a typical service cycle.

In reviewing the signal interaction that takes place during service cycles, the following points should be carefully noted:

- Either the IOP or the Device Controller may drive ED, thus signalling the last valid byte of data exchange (as opposed to control information). Only the

IOP may drive ES, and the Device Controller must disconnect itself after the subcycle in which it sees ES.

- ED and ES may be interpreted in the following manner:

ED	ES	
0	0	More data to follow; Device Controller must generate at least one more RS.
1	0	This is the last data byte, but the Device Controller must generate exactly one more RS to get the Terminal Order, then disconnect itself.
x	1	This is either the last data byte with no Terminal Order to follow, or the Terminal Order itself. In either case, the Device Controller must disconnect after this subcycle.

- The Multiplexing IOP is designed in such a manner that at most, four bytes of data may be exchanged in one service cycle, while the Selector IOP can exchange data indefinitely in one cycle. The Multiplexor IOP generates ED at least once every four bytes, thus ending the service cycle, and may generate ED after any or every byte, thus allowing only one byte of data to be exchanged per service cycle.

Details regarding the information exchanged during Order Out, Order In, and Terminal Order are contained in the following three sections. The function of these cycles, and the differences between Order Out and Terminal Order, may be better understood by reference to the details of bit significance during these operations.

During a service cycle, the DOR and IOR lines are scanned by the IOP only during the first RS of that service cycle. All data exchanges during that service cycle are then of the same type (Data/Order) and in the same direction (Input or Output) as the first byte, except for the Terminal Order, which is unique. However, it is recommended that the Device Controller continue to drive the DOR and IOR lines throughout the entire service cycle, as this facilitates certain test operations.

It should also be noted that the specification of input or output is made by the Device Controller, and the IOP faithfully performs the specified operation. It is thus possible to design a Device Controller that performs an input operation on one service cycle, and an output operation on the next cycle. The IOP reads or writes memory as directed, interleaving these operations if required, so long as each service cycle only requires either a read or a write.

2.6 Order Output

Order Output is the designation assigned to one of the four basic types of service cycles. Its function is to obtain the first (or subsequent) command doubleword from the command list stored in CPU main memory, and to perform the following operations with the various information in the command doubleword:

- Store the new byte count in IOP fast memory.
- Store the new memory byte address in IOP fast memory.
- Store the eight IOP control flags in IOP fast memory.
- Pass the new order along to the Device Controller.

There are two reasons that the Device Controller may request Order Out:

- The device has just been started; the first Order Out is currently obtaining the first command doubleword from memory, and thus deciding what to do (read, write, read backward, etc.).
- Command chaining is required; hence, subsequent command doublewords must be extracted from memory and utilized as described above.

The sequence of operations which allows a Device Controller to ask for Order Out is rather complex. This sequence is:

- When the last data byte (in or out) is processed, the IOP issues a Terminal Order that specifies Count Equal Zero, if and only if data chaining is <u>not</u> called for. If data chaining is required, the Device Controller is never supposed to know about it, or even know the byte count has gone to zero.

- The Device Controller must perform an Order In specifying Channel End. This does not mean the Device Controller disconnects yet.
- In the Terminal Order that follows Order In a bit designates whether or not command chaining is to be performed. (The CC bit is present during every Terminal Order, but should only be inspected by the Device Controller after the Order In that specifies Channel End). If this bit is not set, the Device Controller should disconnect itself, thus requiring another SIO to start up. If the Command Chaining bit is set, the Device Controller should request Order Out on its next service cycle, and begin another operational sequence.

The significance of the data lines during Order Out is as follows:

MMMMMM11	Control
MMMMMM01	Write
MMMMMM10	Read
MMMM1100	Read Backward
MMMM0100	Sense
I0000000	Stop
XXXX1000	Transfer in Channel (never seen by Device Controller)

where: M denotes control bits whose significance is unique to each Device Controller.

I denotes that the Device Controller should issue interrupt request if I = 1.

Stop code should produce the same effect as an HIO instruction.

In SIGMA 5 and SIGMA 7, Order Out undonditionally is followed by a Terminal Order. Thus, during Order Out, the Device Controller should drive ED. The IOP holds ES low. The explanation of the information contained in that Terminal Order is given in Section III-2.8. Timing for Order Out is shown in Figure 3-4.

2.7 Order Input

Order Input is the designation assigned to one of the four basic types of service cycle. Its function is to allow the Device Controller to communicate certain control information to the IOP. This information is placed on the data lines by the Device Controller when it brings up Request Strobe. The Device Controller should also drive ED, and the IOP might hold ES low, thus producing the conditions that lead to a Terminal Order.

The control information placed on the data lines by the Device Controller during Order In has the following significance:

Bit 0 Transmission Error (TE) detected by the Device Controller.

Bit 1 Incorrect Length (IL): the length of a record was incorrect, usually with respect to some physical characteristic of the device (for instance, other that 80 bytes transmitted to the card punch from the IOP).

Bit 2 Chaining Modifier: if this bit has been set, the next Order Out execution causes the IOP to skip one command doubleword (two locations) in the main memory command list.

Bit 3 Channel End: this is an artificial condition that signifies the Device Controller is ready either to initiate command chaining or to disconnect itself if command chaining is not called for in the Terminal Order following Order In. It should be signalled only following the Count Equals Zero in the previous Terminal Order.

Bit 4 Unusual End: this signifies unusual conditions in the Device Controller that require disconnect. The Device Controller should disconnect itself after it issues this notice to the IOP (after it has inspected the Terminal Order that may follow). If the Terminal Order calls for interrupt, the interrupt request should be issued, but the Device Controller should disconnect immediately.

The remaining bits in an Order In have not been assigned any significance. Timing for Order In is shown in Figure 3-4.

2.8 Terminal Orders

A Terminal Order is a data transmission from the IOP to a Device Controller; it may conclude certain other data exchanges. A service cycle may not consist of a Terminal Order by itself, but must involve either Data Out, Data In, Order Out, or Order In first. Order Out is always followed by a Terminal Order in SIGMA 5 and SIGMA 7, although there may be no information the IOP must communicate in that Terminal Order (in which case all the data lines are false).

Data Out and Data In cycles may consist of one, two, three, or four bytes. Since both the IOP and the Device Controller may control ED, either one may determine the number of bytes exchanged in a service cycle. Although a Device Controller may ask for, say, four bytes in a service cycle, it is possible for the IOP to abort operations sooner if it needs to transmit a Terminal Order immediately. The IOP may issue a Terminal Order for any of the following reasons:

- to request that the Device Controller generate an interrupt request;
- to signal Count Equal Zero" to the Device Controller; or
- to signal the Device Controller that an error has been detected (either internally in the IOP, or with respect to the IOP-memory communications, or because of IOP-Device Controller communication error), and the Device Controller should halt. In conjunction with this, the Device Controller may be instructed to ignore the last byte of data.

The bits in a Terminal Order have the following significance:

Bit 0 Interrupt: the Device Controller must respond by generating an interrupt request.

Bit 1 Count Done: this is signalled when the byte count equals zero, if and only if data chaining is not called for. The Device Controller must respond to this condition with another service cycle, specifying Order In, in which Channel End is signalled.

Bit 2 Command Chain: if the CC bit in IOP fast memory is set, the CC bit is true in every Terminal Order. However, the Device Controller should inspect this bit only during the Terminal Order following the Order In during which Channel End was reported to the IOP. At that time, if this bit is reset, the Device Controller should disconnect itself and not issue any more service calls until a subsequent SIO instruction is issued. One exception exists; if bit 0 is also true, the Device Controller should issue an interrupt request and wait for its acknowledgement after disconnecting, and not accept SIO's until an AIO has been issued. If the CC bit is set, command chaining is called for, and the Device Controller should request another service cycle, during which Order Out should be called for to get the next command.

Bit 3 IOP Halt: this signifies that the IOP has detected one of a variety of conditions that inhibit it from successfully completing the required operations for the channel; the Device Controller must disconnect.

Bit 4 Ignore Last Byte: this is signalled only following Data Out or Order Out, when the IOP suspects that the previous byte was incorrect. If possible, the Device Controller should ignore that last byte, although in many cases this may be impossible.

The remaining bits in a Terminal Order have been assigned no significance. A typical service cycle involving Terminal Order is shown, with detailed timing, in Figure 3-4.

2.9 Interrupt Calls and Acknowledgement

A Device Controller is allowed to activate the Interrupt Request (IR) line to the IOP, except during the short interval when anyone's interrupt is being acknowledged by an AIO instruction. The reasons for generating an IR might be strictly internal to the Device Controller, or a manual pushbutton on a device, or a "command" from the IOP (via bit 0 of any Terminal Order) to generate an interrupt, or bit 0 set in a STOP command received during Order Out.

The IOP (CPU) uses the AIO instruction to determine which of many possible Device Controllers or devices has the highest priority interrupt request pending. The timing and signal interaction for AIO are identical to SIO, HIO, TIO, and TDV, except that the function indicator AIO brackets FS, thus distinguishing this instruction from the others, as shown in Figure 3-3.

The Device Controller's required response to AIO is also somewhat different. Priority determination and acknowledgement of FS (both leading and trailing) are exactly identical to the other instructions (the subcontroller handles all this), but the information placed on the data and function response lines by the Device Controller is different. During AIO, the highest priority Device Controller must put its own address on the function response lines FR0 - FR7. It must also put status information on the data lines. This status information is unique to each particular device, and is thus described in the individual specifications for each device.

The HPI (High Priority Interrupt) signal bears the same relationship to IR that HPS does to SC. A device that needs to use HPI must also use IR. In responding to AIO, any device with HPI set gets precedence over one with only IR set. A Device Controller with only IR set can have its interrupt request acknowledged only if there are no Device Controllers that have set IR, and even then only if it is the highest priority Device Controller with IR set.

3.0 DEVICE SUBCONTROLLER

3.1 General

The Device Subcontroller, hereafter simply called Subcontroller, is a set of nine modules that serve a number of functions common to all Device Controllers:

- They provide for manual selection and alteration of Device Controller address.
- They provide priority determination logic for both service calls and interrupt requests, with Subcontroller output signals fed to the Device Controller to indicate when it is selected.
- They provide for power up-down controls, so that a Device Controller may be powered down without disconnecting cables, and other Device Controllers on the line may still function properly.
- They provide the logic for certain common areas, such as a "Service Connect" control flip-flop.

3.2 Subcontroller Component Parts

The nine modules comprising the Subcontroller serve the following functions:

AT10 Cable Receiver , to which one of the cables from the IOP connects

AT11 Cable Driver/Receiver, to which one of the cables from the IOP connects

AT12 Cable Driver, to which one of the cables from the IOP connects

AT17 Cable Driver/Receiver and Power On-Off Relay, to which the input and output priority determination cable connects (occupies two module positions)

LT22 Special Logic Module, containing some of the priority determination logic and various other Subcontroller functions

LT23 Special Logic Module, containing some of the priority determination logic, the service-connect flip-flop (FSC), TSH, and TTSH

LT24 Special Logic Module, containing function response line buffers with three-way OR's and other Subcontroller functions

LT25 Special Logic Module, containing service call latch circuits, data line (received) inverters, a toggle switch (to indicate on-line or off-line) and other Subcontroller functions

LT26 Special Logic Module, containing eight toggle switches for Device Controller address selection, and two 4-bit comparators to compare this selected address with IOP output address during SIO, HIO, TIO, and TDV.

An appendix to this manual shows the positions within a chassis where the Subcontroller must be located when it is used. This will allow Customer Service to readily locate the Subcontroller and check its operation when necessary in any standard piece of equipment.

3.3 Subcontroller Usage

The Subcontroller must be used in any piece of equipment designed to connect to the IOP, unless all of the following conditions exist:

- The equipment is not intended to be a standard company product.
- The equipment utilizes two or more Multiplexor IOP channel addresses which might effectively share certain Subcontroller logic. The utilization of the two portions of the equipment corresponding to the two addresses thus must be interdependent and non-simultaneous.
- Strong economic advantage can be shown for so doing.

3.4 Subcontroller Summary Description

Various standard documents, listed in the appendixes, show the detailed logic structure of the Subcontroller. The explanation here will discuss only the interface signals between the Subcontroller and the remainder of the Device Controller and their usage.

These signals are:

SWA0 - SWA7	Switch Outputs: used for manual channel address selection.
DCA	Device Controller Address: true when switch settings SWA0 - SWA7 equal data line receivers DA0R - DA7R (DCA-bar also provided).
TTSH	OR-ing of function indicators for SIO, HIO, TIO, and TDV (TTSH = (SIOR + HIOR + TIOR + TDVR)).
TSH	OR-ing of function indicators for SIO, HIO, and TIO, with DCA true (TSH = (SIOR + HIOR + TIOR) · DCA).

The Subcontroller supplies FSL to the line. FST must be supplied to the Subcontroller by the Device Controller. For simple devices that involve no delay, Function Strobe may be tied right back to FST.

The Subcontroller supplies, in addition, the gating on function response line buffers for TDV (one set of eight points) and SIO, HIO, and TIO (a second set of eight points).

The Subcontroller provides latch circuits to receive service and interrupt requests from the Device Controller and pass them on to the IOP. The signals from the Device Controller must be held up until they are properly acknowledged by the IOP, since the latch provided by the Subcontroller is only applicable during the acknowledgement itself. These requests are:

CIL	Interrupt Request from the Device Controller to the Subcontroller.
CIH	High Priority Interrupt Request from the Device Controller to the Subcontroller (CIL must also be issued when CIH is issued).
CSL	Service Request from the Device Controller to the Subcontroller.
CSH	High Priority Service Request from the Device Controller to the Subcontroller (CSL must also be issued when CSH is issued).

The Subcontroller handles internally all priority determination, removing that burden from the Device Controller itself. The signal BSYC is generated during service and interrupt acknowledgements to indicate that this device has the highest priority and is acknowledging.

Each Subcontroller also has a service-connect flip-flop (FSC), which is set by the Subcontroller at the trailing edge of Acknowledge Service Call (gated with FS), and remains set until ES is sensed. Thus, each Device Controller may issue Request Strobes only while its FSC is true.

The receiving circuit outputs from the cable receivers are all available to the Device Controller, as are all the input driving points to the cable drivers. However, it should be noted that it is not possible to feed a signal directly to a cable driver input which is also fed by a Subcontroller signal; this should never be necessary if the Subcontroller is properly utilized.

The Subcontroller provides the entire power on-off logic, allowing power to any Device Controller to be removed without affecting the IOP bus adversely. All data lines are also inverted and provided as outputs from the Subcontroller. An appendix to this manual lists the available unit loads from each Subcontroller output signal.

4.0 DIFFERENCES ON SIGMA 2

The differences between the IOP interface on SIGMA 2 and those of SIGMA 5 and SIGMA 7 relate only to the SIGMA 2 reaction to certain stimuli, and not to the appearance or timing of interface signals. Thus, the differences that do exist should not affect the external equipment designer in any way. This section is provided for information only.

The primary difference is in the method of accomplishing command chaining. As it was noted before, in SIGMA 5 and SIGMA 7 a Terminal Order is delivered to the Device Controller; this Terminal Order specifies zero byte count. The Device Controller must come back with an Order In that specifies Channel End. In the Terminal Order that follows Channel End, the IOP specifies command chaining if it is to occur, and the Device Controller either asks for Order Out (and the new Order), or disconnects (if the command chaining bit is true in the Terminal Order following Order In). In SIGMA 2, command chaining is never specified to the Device Controller. Thus, since the command chaining bit is false, the Device Controller must disconnect. Another SIO is required to re-initiate operations. Command chaining is thus accomplished in SIGMA 2 with a sequence of SIO's, with complete disconnect of the Device Controller after the operations associated with each SIO.

5.0 INTERFACE SIGNAL LINES AND CABLING

Appendix C gives module pin numbers for these cables. The designer must ground inputs to drivers for signals that are not to be driven, or for response bits that are not used by a particular Device Controller.

The cables are:

Cable #1 connects to the AT12 cable driver module. This cable transmits function response signals 0-7 (FR0D - FR7D), Request Strobe (RSD), Input/Output Request (IORD), Function Acknowledge Leading (FSLD), Function Acknowledge Trailing (FSTD), and Interrupt Call (ICD).

Cable #2 connects to the AT11 cable driver/receiver module. This cable transmits and receives Data Lines (DA0D - DA7D and DA0R - DA7R), Data Parity (DAPD and DAPR), End Data (EDD and EDR), Parity Check (PCD), Data/Order Request (DORD), and Service Call (SCD and SCR).

Cable #3 connects to the AT10 cable receiver module. This cable receives I/O Reset (RSTR), the one-megacycle clock (CL1R), End Service (ESR), Request Strobe Acknowledge (RSAR), six function indicators (SIOR, HIOR, TIOR, TDVR, AIOR, and ASCR), and Function Strobe (FSR).

Cable #4 connects to the AT17 cable plug module. This cable receives and transmits High Priority Interrupt (HPID and HPIR), High Priority Service (HPSD and HPSR), and Busy (BSYD and BSYR). It receives Available Input (AVIR) and transmits Available Output (AVOD).

6.0 LOGIC DESIGN OF DEVICE CONTROLLERS (DESIGN TIPS)

6.1 General

Almost all Device Controllers will use the Device Subcontroller described in section III-3, with additional information in Appendix F and in the reference specification drawings for the Subcontroller. Certain other functions are required in the remainder of the Device Controller, regardless of the unit's function. This section will show typical and recommended implementations for various Device Controller functions in this category.

In the logic diagrams shown throughout the remainder of this section, certain logical signal names are used. These have the following significance, where an "N" preceding a signal name denotes its negation:

B4	Byte 4 for Device Controllers designed for 4-byte service cycles
BSYC	Subcontroller priority determination signal indicating that this Device Controller has the highest priority (true during FSR)
CHEND	Channel End storage
CLIR	1-megacycle clock from IOP
DAiR	Data bit, i receiver
DCA	Device Controller address (Subcontroller signal)
EDD	End Data driver
EDR	End Data receiver
ESR	End Service receiver (IOP to Device Controller)
FSC	Subcontroller service-connect flip-flop
FSR	Function Strobe receiver (IOP to Device Controller)
HIOR	HIO function indicator
OIN	Order In control flip-flop
OOUT	Order Out control flip-flop
RSAR	Request Strobe Acknowledge receiver (IOP to Device Controller)
RSD	Request Strobe driver (Device Controller to IOP)
RSTR	I/O Reset signal from IOP

SIOR	SIO function indicator
START	Start control flip-flop
TO	Terminal Order control flip-flop
TSH	Subcontroller signal for function indicator (SIOR + HIOR + TIOR)
UEND	Unusual End storage
WANTI	Device Controller Wants Interrupt control flip-flop

6.2 "START" Flip-Flop and Control Flip-Flop Initialization

The START control flip-flop signifies that the Device Controller has been started by an SIO instruction, and may therefore issue Request Strobes and engage in Data/Order exchanges with the IOP. It should be noted that an SIO instruction must be rejected if an interrupt is pending (not yet acknowledged). The START state can be reset under the following conditions, as shown in Figure 3-5 on the following page:

- unconditionally by an HIO instruction;
- by a Terminal Order injunction for IOP Halt;
- following Unusual End reporting to the IOP via Order In;
- by the absence of the command chaining bit in the Terminal Order that follows the Order In in which Channel End was reported; or
- by a "STOP" order given via Order Out.

In some devices it is inconvenient or undesirable to cause unconditional termination of operations with an HIO. In such devices as the card reader, the HIO is not allowed to cause any action until the end of the current card. This may often be accomplished by causing the HIO instruction to set an auxiliary flop (may be the same one that remembers incorrect length) , which in turn causes Unusual End, Channel End, and resetting of START at the same time as device end. To accomplish this, the input term to the reset logic in Figure 3-5 (designated HIOR) should be wired to ground.

Two reset signals are also needed for the control flip-flops throughout a Device Controller: one which unconditionally resets everything (the I/O reset signal from the IOP), and another which resets only those control flip-flops that should not be active unless the Device Controller has been started. NSTART accomplishes the second of these functions, whenever it may be needed.

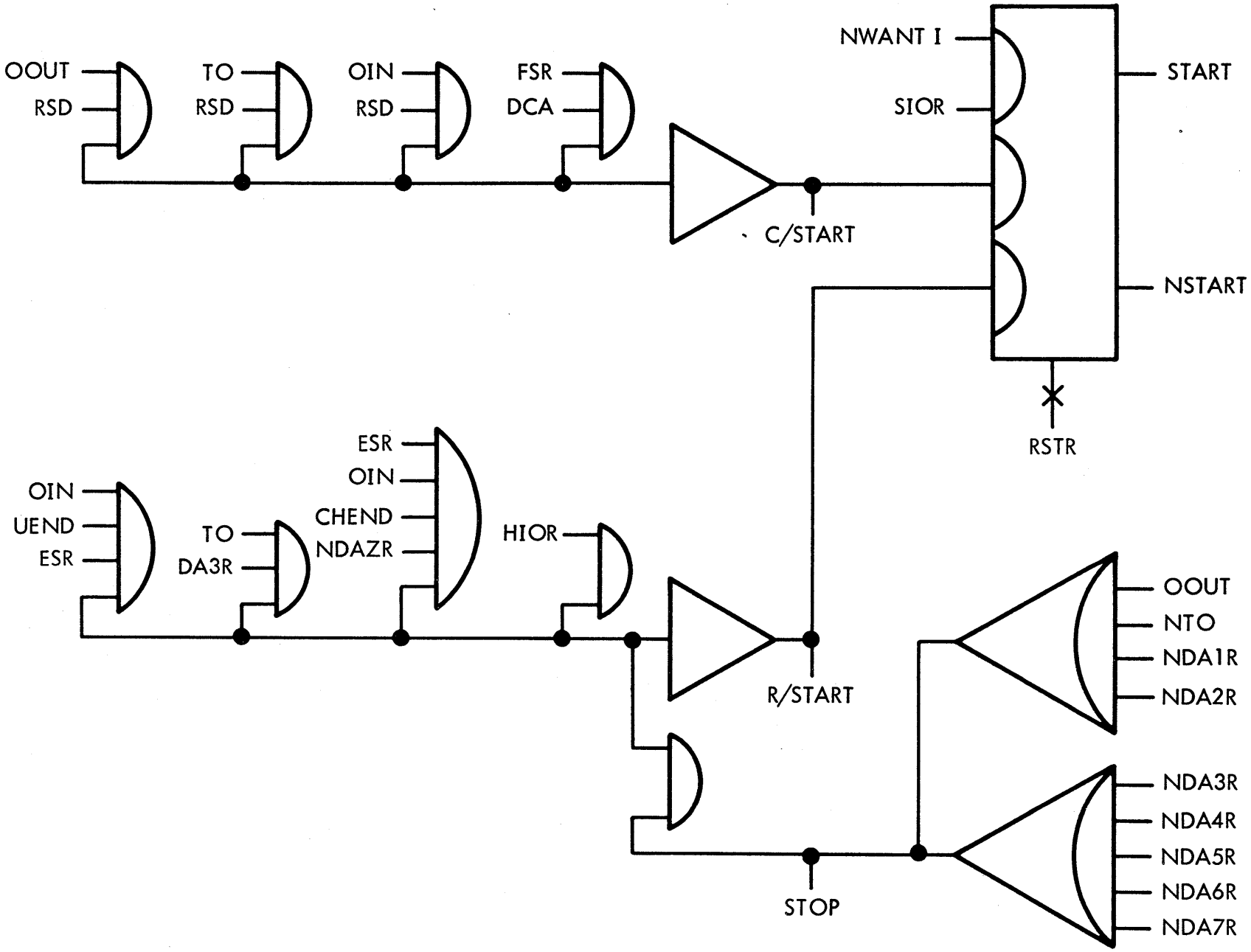


Figure 3-5. "START" Flip-Flop and Control Flip-Flop Initialization

6.3 Terminal Order Control Flip-Flop

A control flip-flop may be used to define the period during which a Terminal Order may take place. The actual Terminal Order is defined by (RSD · TO).

Figure 3-6 shows this logic.

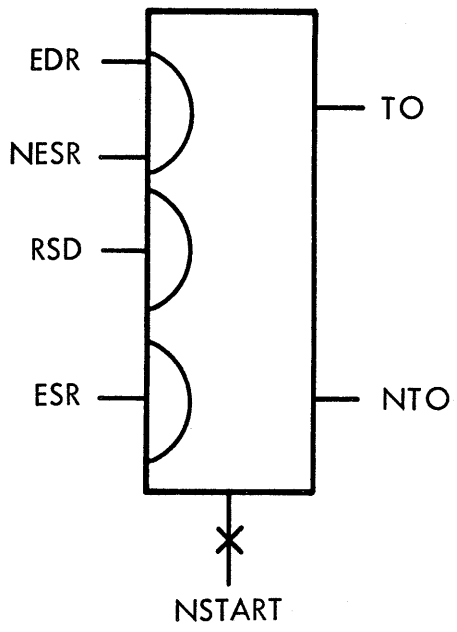


Figure 3-6. Terminal Order Control Flip-Flop

6.4 Order Out Control Flip-Flop

A flip-flop may be used to remember the conditions under which an Order Out cycle is needed. Order Out cycles should take place under the following circumstances:

- An SIO instruction has been accepted by the Device Controller.
- The command chaining bit is set in the Terminal Order that follows the Order In in which Channel End was reported to the IOP.

The logic shown in Figure 3-7 on the following page, along with the proper flip-flop resetting, accomplishes this.

The clock term is required to allow resetting of OOUT in the event that the Device Controller is disconnected before the Order Out service cycle is executed (for instance, by an HIO instruction). Incorrect Length might be such a reason for generating Unusual End.

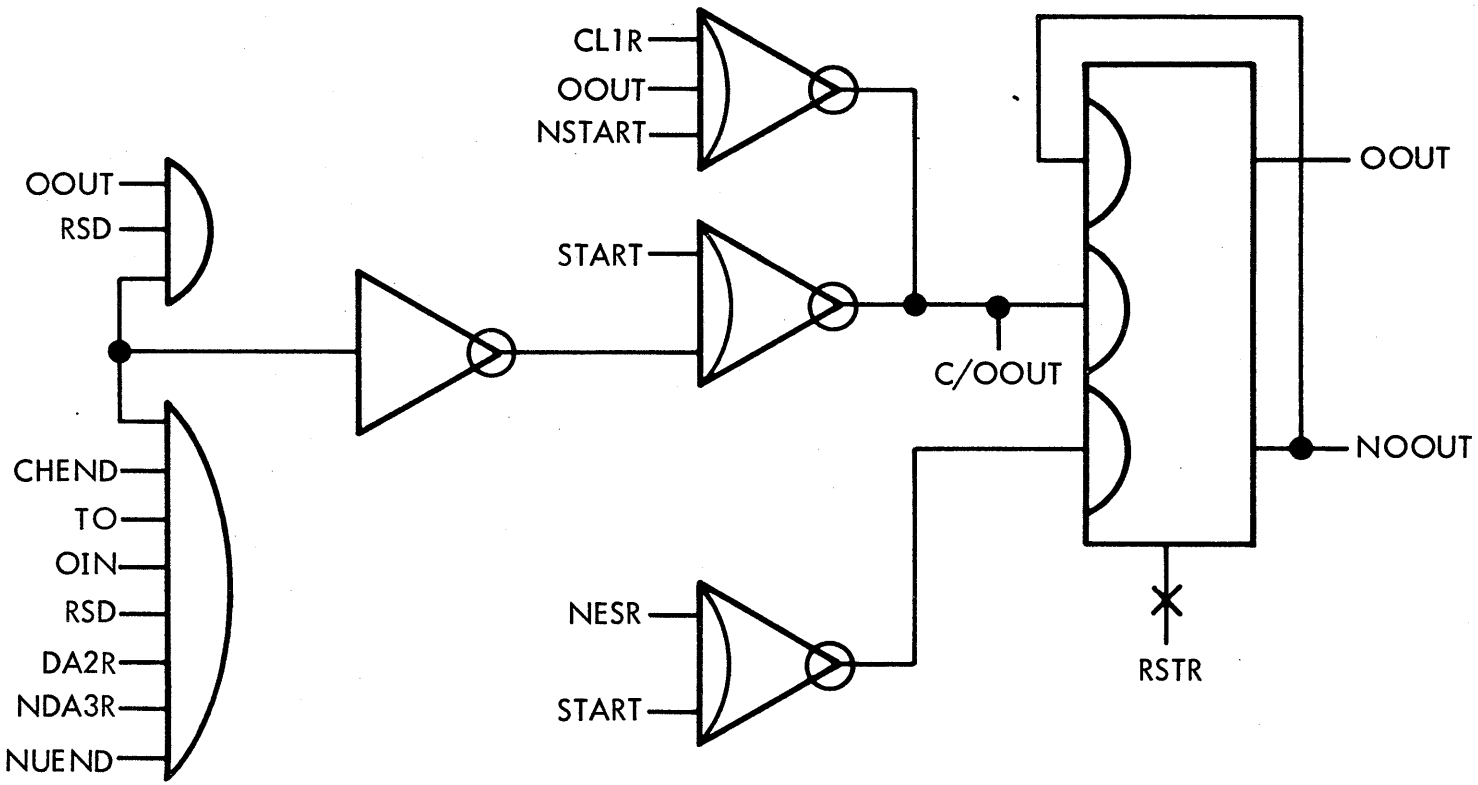


Figure 3-7. Order Out Control Flip-Flop

6.5 Order In Control Flip-Flop

A flip-flop may be used to remember the conditions under which an Order In cycle is needed. Order In cycles should take place under the following circumstances:

- A condition requiring Unusual End has been detected in the Device Controller.

- A Terminal Order was received in which Count Equals Zero was specified. The Order In is now required to specify Channel End to the IOP, and to determine whether or not command chaining is required (in the Terminal Order following that Order In). The Device Controller may also report the chaining modifier and/or Incorrect Length in that Order In.

The logic shown in Figure 3-8 on the following page, along with proper flip-flop resetting, accomplishes this.

The blank gate shown in Figure 3-8 might be used for other conditions in the Device Controller requiring Channel End, or Order In for other reasons. This occurs in some of the standard system junction boxes.

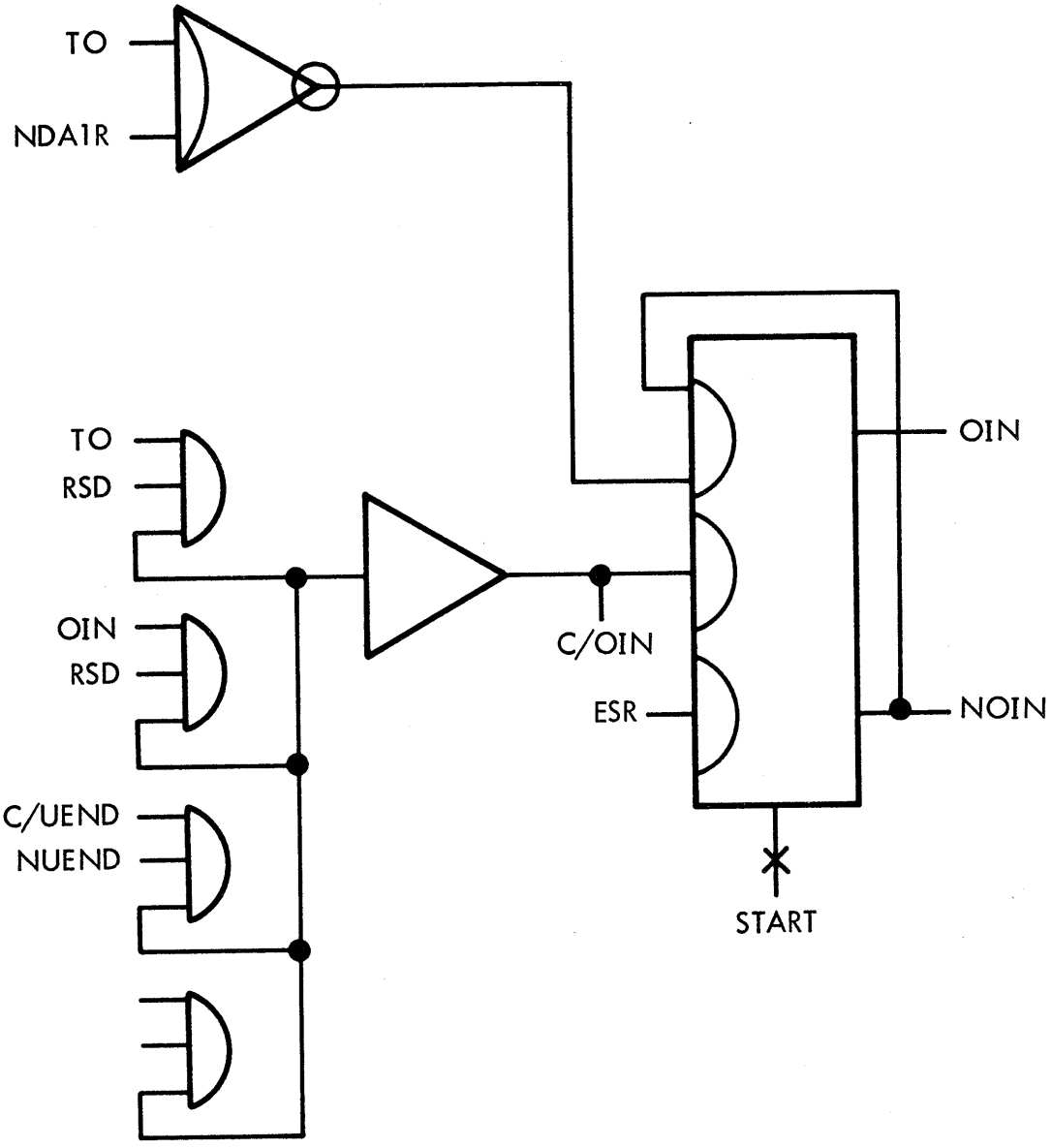


Figure 3-8. Order In Control Flip-Flop

6.6 "WANT SERVICE" Control Flip-Flop

When a Device Controller wants service from the IOP, it must raise the line designated CSL into the Subcontroller. This signal is not latched in the Subcontroller until the ASC cycle actually begins, so that dropping it before that time causes the Service Request line to be dropped. Therefore, a flip-flop is generally required in the Device Controller to store the conditions that define a service requirement.

The conditions under which a Device Controller might want service are as follows:

- following the Order Out cycle, providing that if a Terminal Order follows the Order Out cycle, it does not specify IOP Halt;
- when ED occurs during a multiple-byte data service cycle before all bytes required by the Device Controller have been taken (for example, if the IOP crosses a word boundary in exchanging data);
- under other conditions that are a function of the particular Device Controller (in many cases this might involve delaying service requests after Order Out); or
- to execute a required Order Out or Order In cycle.

It should be noted that a Device Controller should not ordinarily raise CSL during its own service cycle, since an as-yet-undetected Terminal Order at the end of that service cycle might negate the requirement for additional service. Figure 3-9 shows the implementation to handle these various cases; the blank gates would handle anything that is a function of a particular Device Controller, as described above. Figure 3-9 also shows how a Device Controller might refuse to issue service calls if any other device had an active service call, but would latch up its own call once it was issued (the two inverters in the lower right-hand corner of the figure).

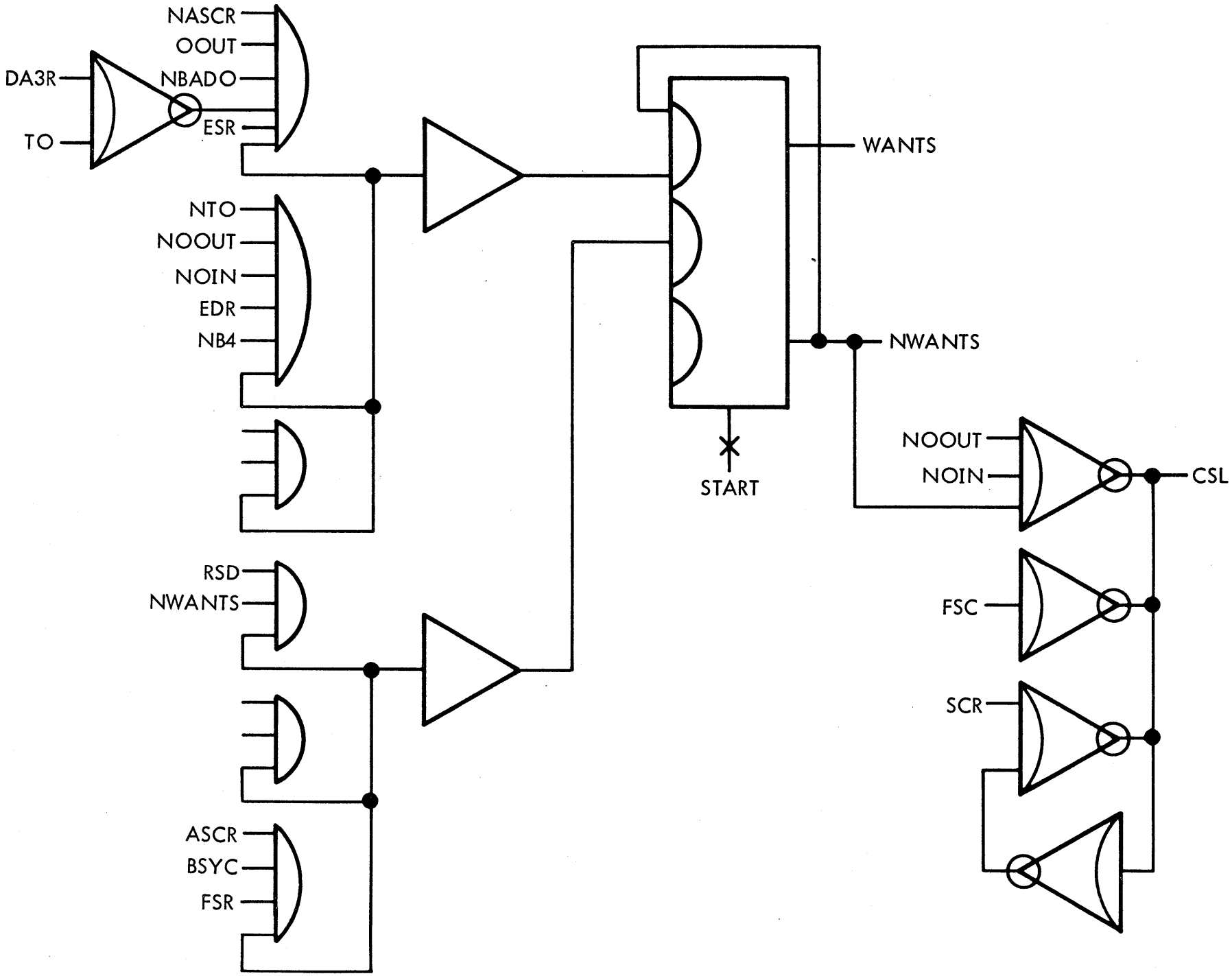
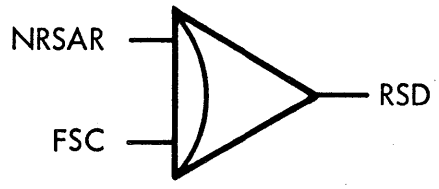


Figure 3-9 "WANT SERVICE" Control Flip-Flop

6.7 · Request Strobe and Acknowledgement

As mentioned before, FSC is the service-connect flip-flop in the Subcontroller. It is automatically reset at the trailing edge of the Request Strobe during which ES is true. Therefore, RS and RSA may be interlocked in the Device Controller in the following manner:



Using the scheme above, IOP/Device Controller interaction for data exchanges proceeds as fast as the IOP and the various cable delays permit. However, valid data may be on the bus for only a very short time (perhaps less than 400 nanoseconds). This is adequate to clock the data into a dc or ac buffer, using two-side loading for dc or trailing edge of RSD for ac loading. There are some devices for which a byte buffer in the Device Controller could be eliminated if data could be guaranteed valid to pass on to the device for only 500 to 600 nanoseconds. This goal can be attained, where the economics of a situation justify it, by the scheme shown in Figure 3-10. However, an extra delay of from 0.6 to 1.6 microseconds is introduced into the IOP/Device Controller interaction cycle, thus reducing IOP bandwidth slightly. The advantages and disadvantages of this procedure should be weighed for each individual unit. In general, introducing a delay any greater than that produced by this method is not permitted. It should be noted that the effect of reducing IOP bandwidth is to increase the probability of rate over-run errors on other active devices.

In Figure 3-10, X represents the conditions that must be true at the leading edge of RSD to cause the "stall" on that service subcycle. The best strobe for outputting data to the device is (RSAR · STALL), which is at least 500 nanoseconds in duration (with a maximum only slightly more).

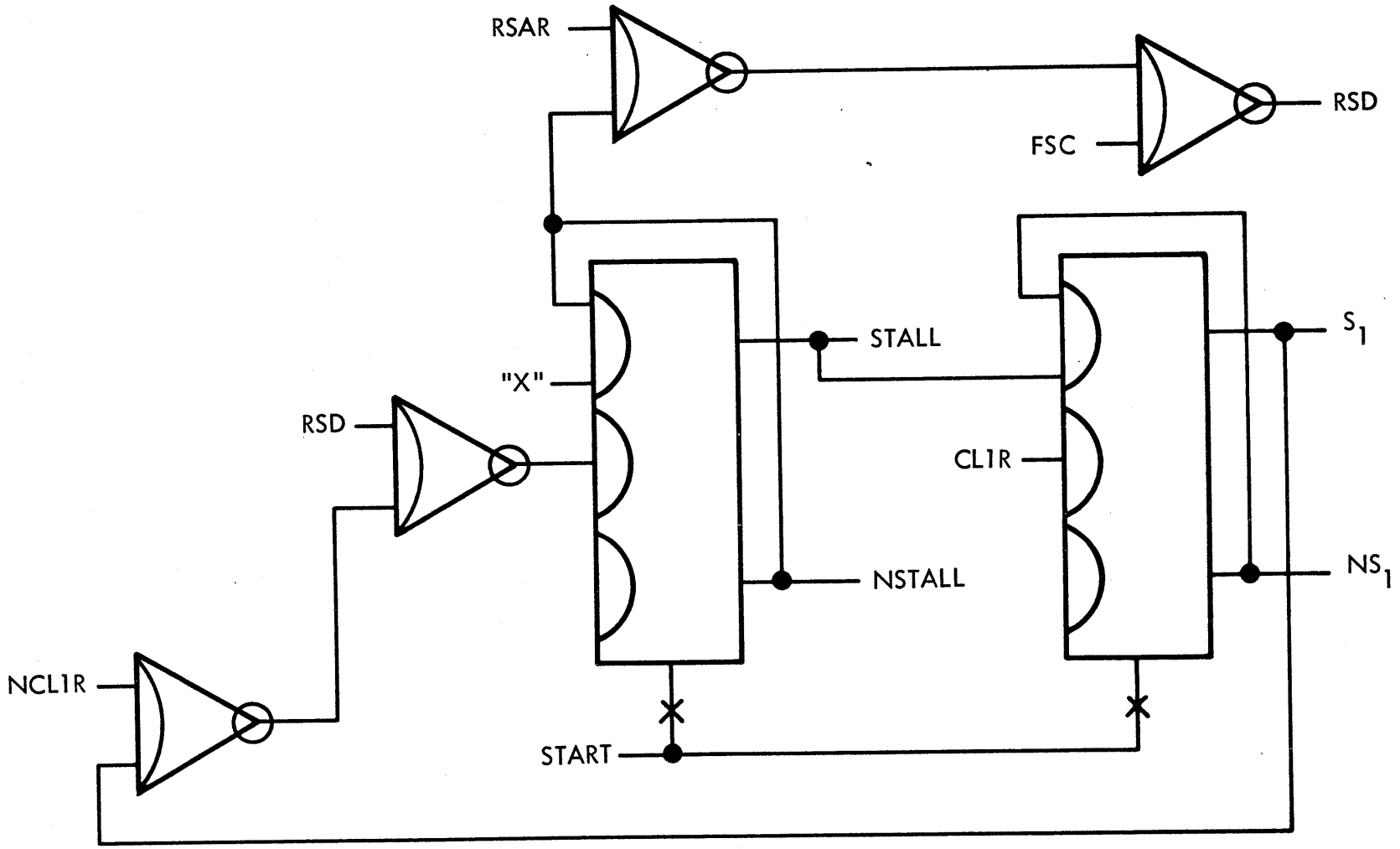


Figure 3-10. Request Strobe and Acknowledgement

6.8 Quantity of Data Bytes per Service Cycle

On the Multiplexor IOP, a Device Controller may control whether it is to exchange one, two, three, or four data bytes with the IOP during each service cycle. A three-byte-oriented Device Controller is unlikely because correlation with main memory cells (on the part of the programmer, in setting up data tables) may prove quite unwieldy. A device designed to exchange more than one byte per cycle might, nonetheless, be permitted by the IOP to exchange only one byte. This could happen as a result of an IOP error, but it could also result from the IOP crossing a word boundary in its interaction with memory.

A one-byte-oriented Device Controller should drive ED in conjunction with every RS. A Device Controller that is oriented towards some greater number of bytes should drive ED only in conjunction with the RS associated with the last byte it wishes to receive. A four-byte Device Controller must therefore have a byte counter similar to the one shown in Figure 3-11, and should drive ED as shown. It should be noted that a one-byte device does not need some of the terms on the WANTS control flip-flop inputs. Those terms are intended only to handle "interrupted" service cycles.

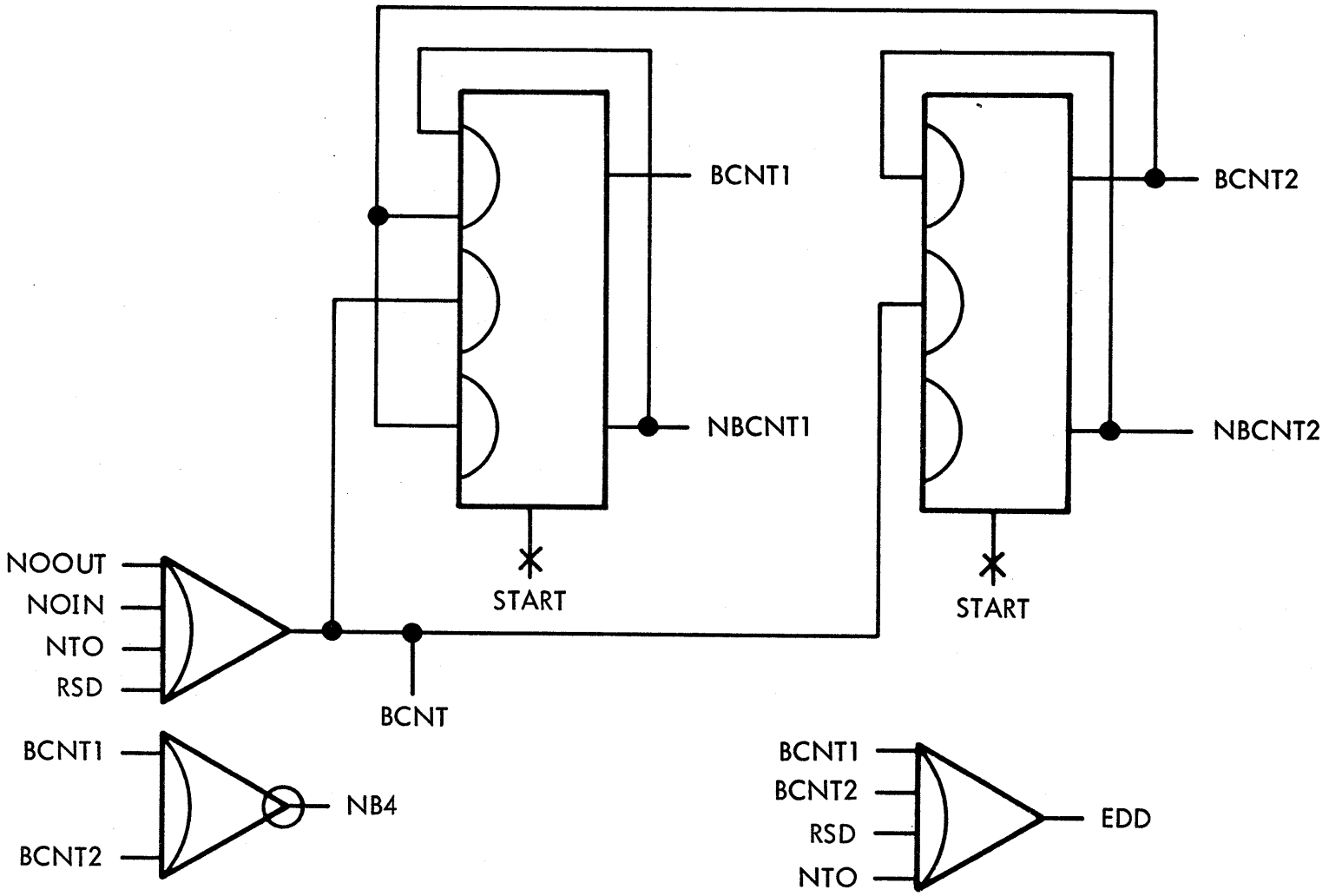


Figure 3-11. Typical Byte Counter

6.9 Channel End

The Channel End state, which may be signalled to the IOP in an Order In, is an artificial condition which signifies that if command chaining is supposed to take place, now is the time when the Device Controller will do so, by initiating an Order Out cycle. Channel End occurs when the Device Controller has been informed via a Terminal Order that Count Equals Zero (without IOP Halt being specified). The Device Controller then tells the IOP that the Channel End condition has occurred, and inspects the Terminal Order which follows that Order In to see if command chaining is specified. If so, Order Out is initiated; if not, the Device Controller is disconnected (START is reset). The logic in Figure 3-12 shows how this may be accomplished.

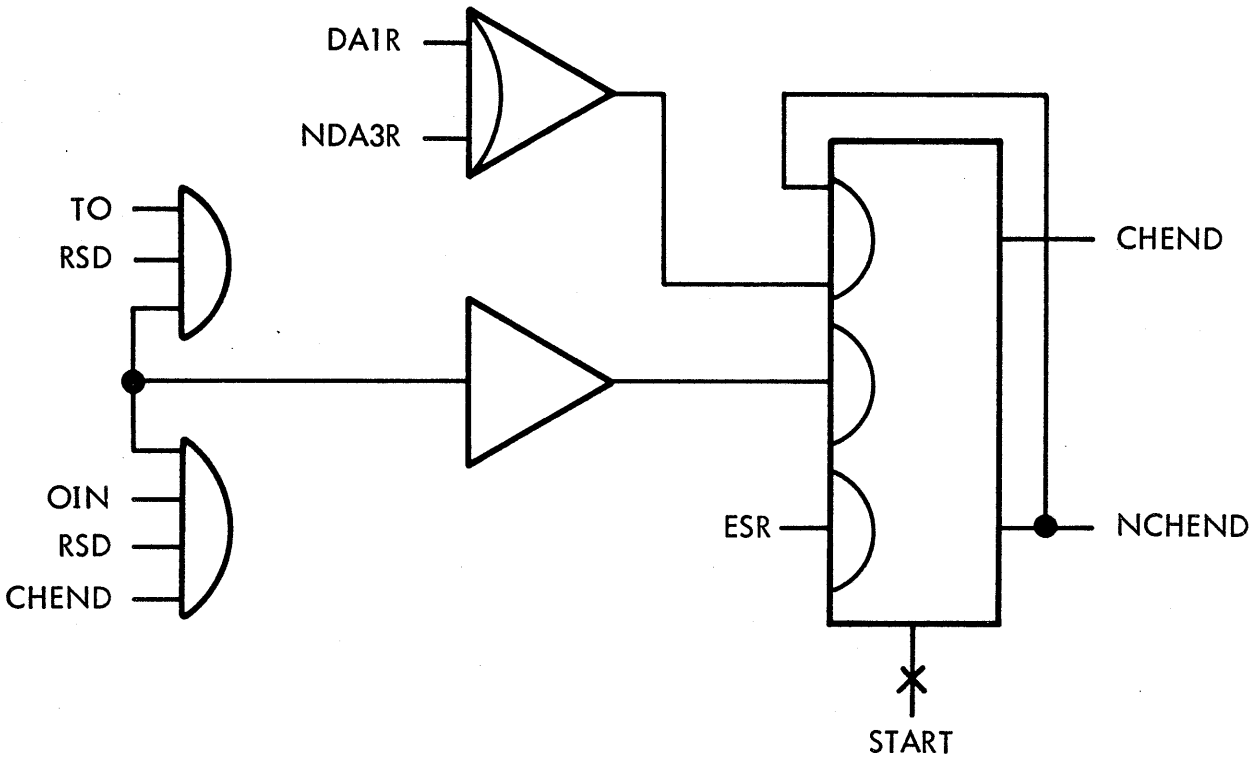


Figure 3-12. Channel End

6.10 "Bad Order" Detection

In Device Controllers where some of the possible orders that may be transmitted during Order Out are either not used or not permitted, the Device Controller should check for these bad orders, and generate Unusual End if they occur, as described in section III-6.11. This detection is quite simple. In Figure 3-13, X represents the decoding of the data lines for those orders considered "bad".

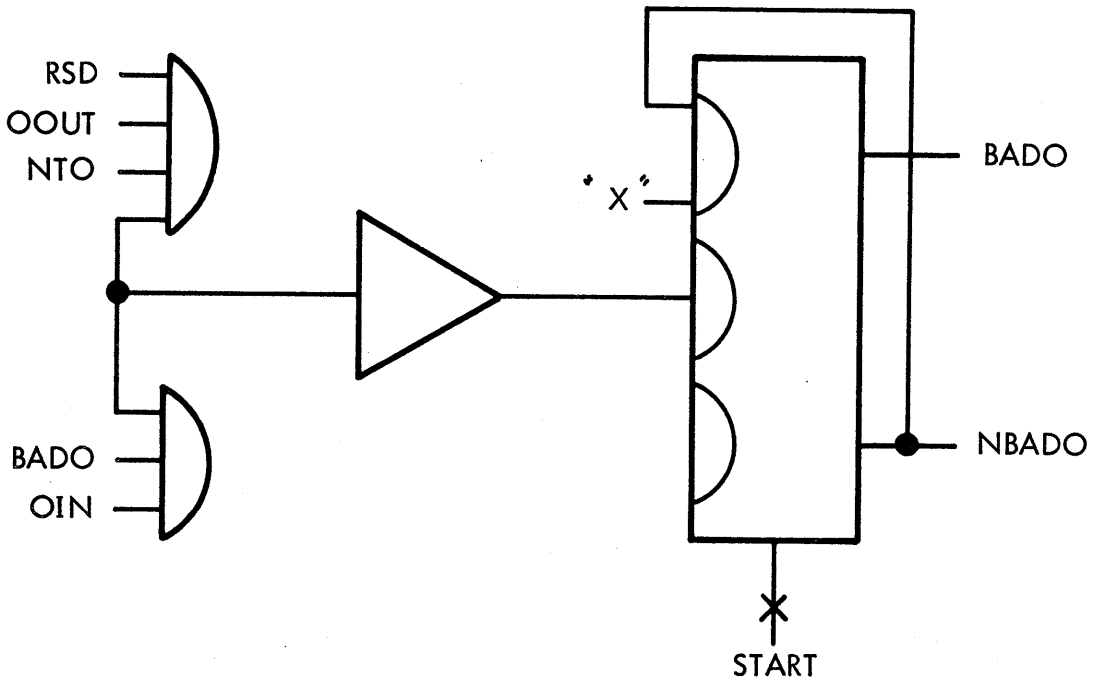


Figure 3-13. "Bad Order" Detection

6.11 Unusual End

The Unusual End condition is produced when some state occurs that inhibits or makes invalid further usage of the Device Controller. If a flip-flop (as shown in Figure 3-14) is set, it will also request an Order In cycle, and cause disconnect of the Device Controller (reset START) after that Order In has been executed. In Figure 3-14, X represents the conditions in the Device Controller which are to cause Unusual End. It should be noted that the flip-flop is not reset until a subsequent SIO, TIO, or HIO instruction occurs.

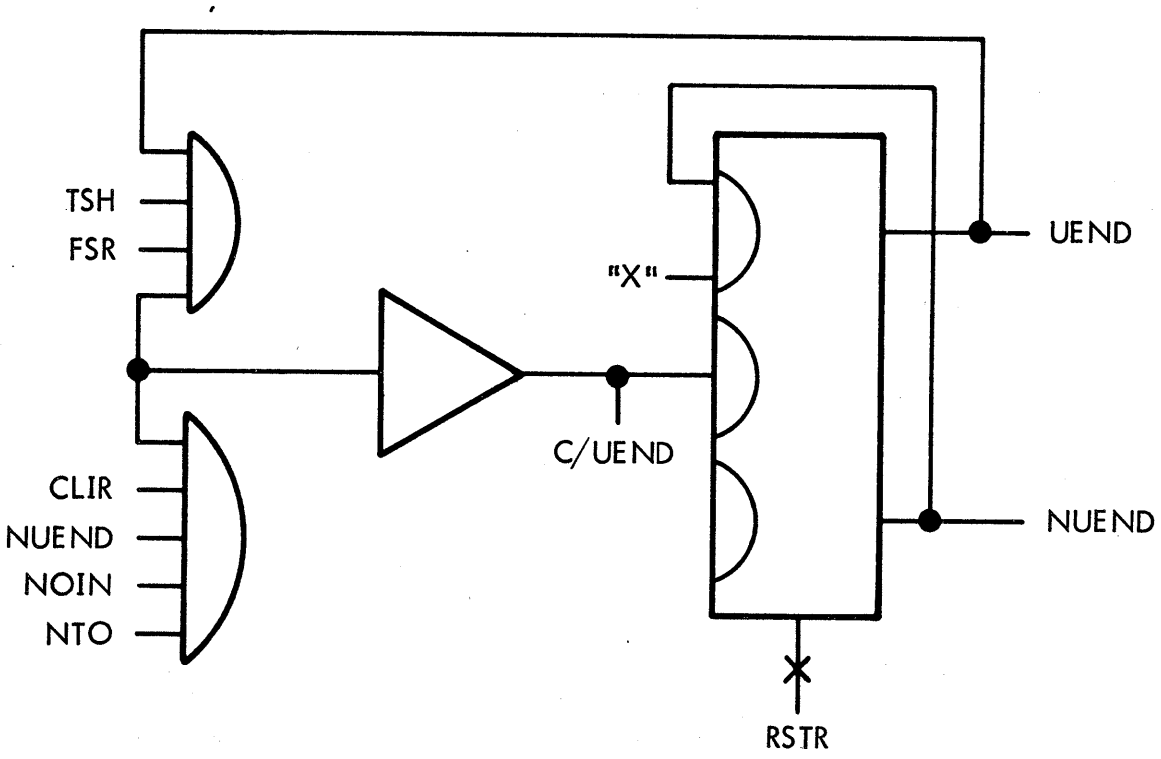


Figure 3-14. Unusual End

6.12 "WANT INTERRUPT" Control Flip-Flop

There are only two standard conditions to which a Device Controller's response must be an interrupt request. These are:

- a Terminal Order with data line zero true
- a "STOP" Order Out with bit zero true

There may also be internal Device Controller conditions which require an interrupt request, including a pushbutton on the device (or on the front panel of the controller for such items as analog equipment controllers).

Figure 3-15 shows a typical implementation that accomplishes the setting of the WANTI control flip-flop. It should be noted carefully that WANTI is one of the few control flip-flops that must not be reset by START; it is reset only when acknowledged. Furthermore, SIO instructions must be rejected by the controller so long as an interrupt is pending (WANTI set).

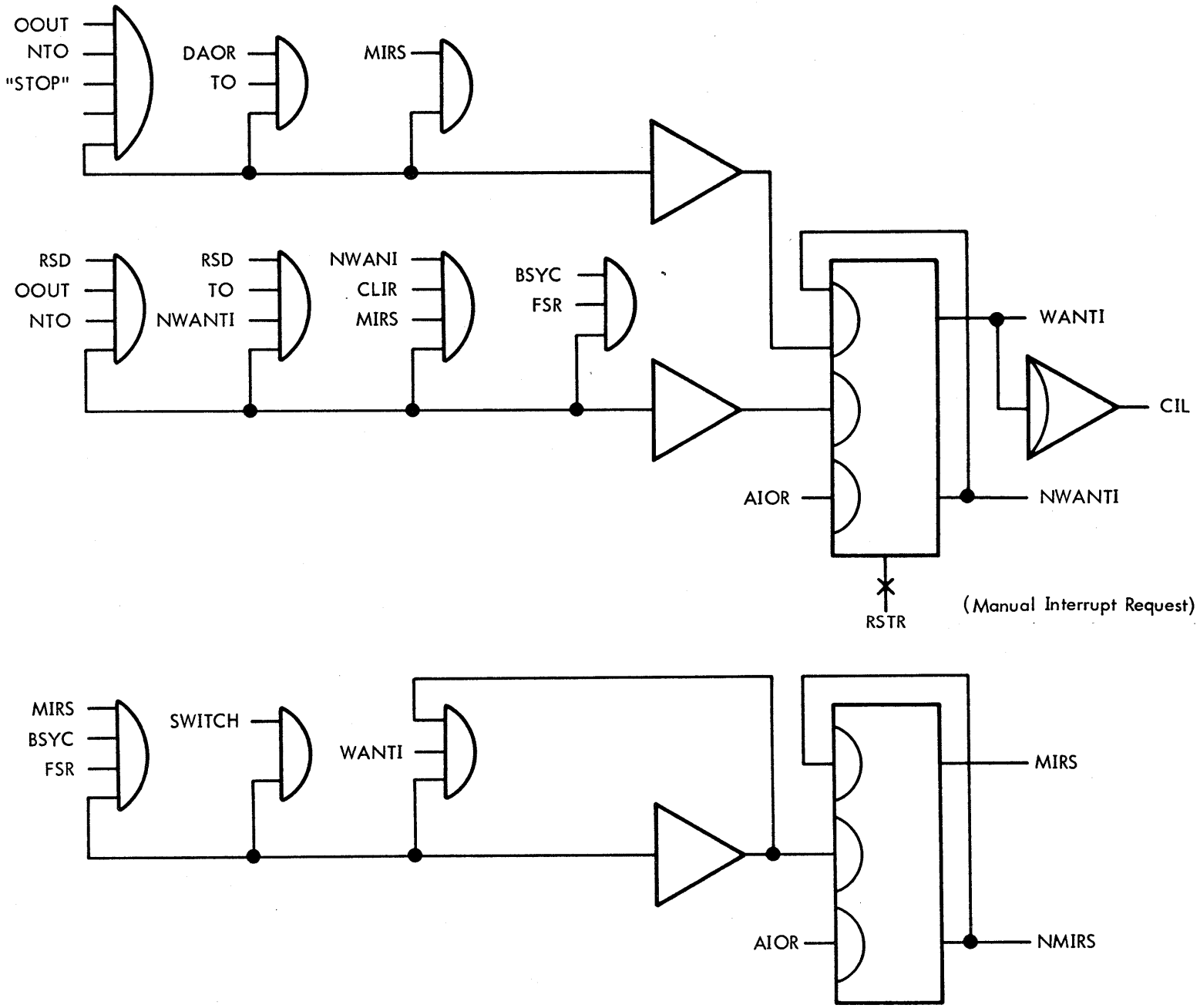


Figure 3-15. "WANT INTERRUPT" Control Flip-Flop

6.13 State Flip-Flops

The examples given in this section of the manual for proposed implementation are only one method. Another technique in common use for standard peripheral equipment, which is often more complex in terms of the number of possible states of the controller than systems equipment, is to have some small number of control flip-flops that define the state of the controller. In general, there can be a configuration of the state flip-flops corresponding to each of the following conditions, as well as others:

- Order In
- Order Out
- Data In or Out
- Waiting for device "go" signal
- Waiting for device "terminate" signal
- Waiting for device "operational" signal (often generated manually)

The approach to be taken for any particular unit must be weighed by the designer according to the particular equipment requirements and the relative cost of implementation of the two methods.

Figure 3-16 shows the possible states of a Device Controller that utilizes the type of control described here, including all possible "next states" and the conditions that cause them.

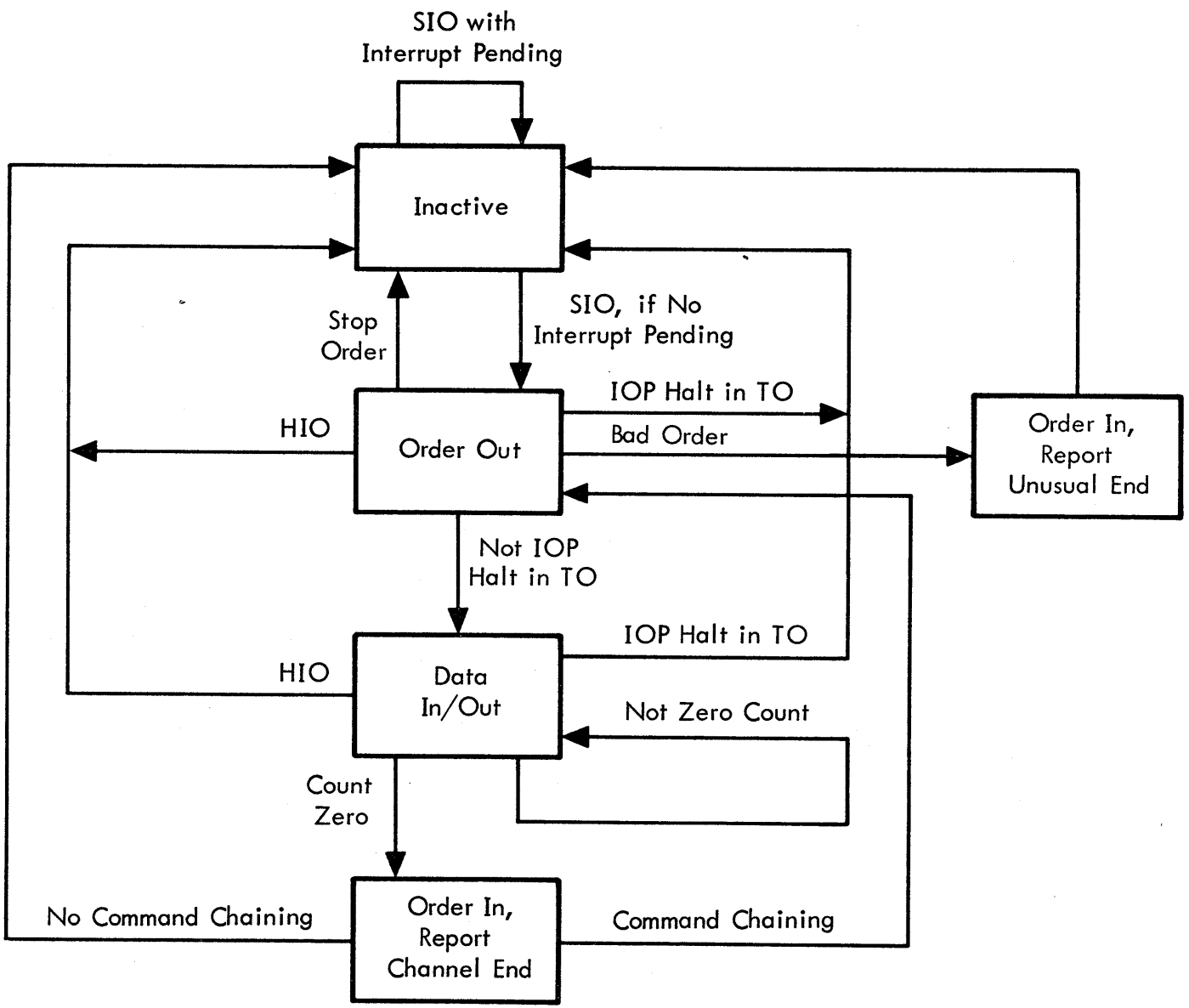


Figure 3-16. State Flip-Flops

APPENDIX A
CABLE CHARACTERISTICS

Standard cable utilized throughout the SIGMA system for inter-communication between the CPU, memories, IOP's, and Device Controllers consists of 14 shielded wires (transmission lines) as specified under SDS Part No. 101787. The nominal wire characteristics are:

Characteristic Impedance:	33 ohms
dc Resistance (Center Conductor):	23 milliohms per foot
dc Resistance (Shield):	10 milliohms per foot
Inductance:	50 nanohenry per foot
Capacitance:	50 picofarad per foot
Signal Delay:	1.4 nanoseconds per foot

APPENDIX B

SUMMARY OF BIT CODINGS FOR VARIOUS DATA EXCHANGES

Function		L	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
SIO, HIO, TIO, TDV		DA	Device Controller Address							
SIO, HIO, TIO, TDV		FR	Interrupt Pending	00 = D Ready 01 = D Not Operational 10 = D Unavailable 11 = D Busy		Device Automatic	Device Unusual End	00 = DC Ready 01 = DC Not Operational 10 = DC Unavailable 11 = DC Busy		---
TDV		FR	Rate Error	---	---	---	---	---	---	---
AIO, ASC		FR	Device Controller Address							
Order In		DA	Trans. Error	Incorrect Length	Chaining Modifier	Channel End	Unusual End	---	---	---
Terminal Order		DA	Interrupt	Count Done	Command Chain	IOP Halt	Ignore Last Byte	---	---	---
Order Out (even command word bits 0-7)	Control Write Read R. Back Sense TIC Stop	DA	M M M M M X I	M M M M M X 0	M M M M M X 0	M M M M M X 0	M M M 1 0 1 0	M M M 1 1 0 0	1 0 1 0 0 0 0	1 1 0 0 0 0 0
Odd Command Word Bits 0-7			Data Chain	Interrupt Count =0	Command Chain	Interrupt Channel End	Halt Transmission Error	Interrupt Unusual End	Suppress Incorrect Length	Skip

APPENDIX C

SUMMARY OF CABLE SIGNALS

Cable	DC J1	DC J2	DC J3	DC J4	RWD J1	RWD J2	RWD J3	RWD J4			
Module Type	AT12	AT11	AT10	AT17	AT11	AT11	AT11	AT11			
Conn. Loc.					23S	27Q	30P	29N			
Module Pin				(PR1)							
1	FR6D	DA6D			DB1D	DB15D	DB29D	△			
2	FR7D	DA7D			DB0D	DB14D	DB28D	△			
3	FR4D	DA4D			DB3D	DB17D	DB31D	△			
4		DA6R	CL1R	HPSR	DB1R	DB15R	DB29R	A03R			
6		DA7R	RSTR	HPIR	DB0R	DB14R	DB28R	A02R			
8		DA4R	RSAR	AVIR	DB3R	DB17R	DB31R	A04R			
9	FR5D	DA5D			DB2D	DB16D	DB30D				
10		DA5R	ESR	BSYR	DB2R	DB16R	DB20R				
12	FR3D	DA3D			DB4D	DB18D	△	△			
13		DA3R	SIOR		DB4R	DB18R	RFSR	A00R			
15	FR2D	DA2D			DB5D	DB19D	RFSAD	△			
18		DA2R	HIOR		DB5R	DB19R		A01R			
19	FR1D	DA1D			DB6D	DB20D	△	△			
20		DA1R	TIOR		DB6R	DB20R	RWDR	A08R			
22		DA0R	TDVR		DB7R	DB21D	△	△			
23	FR0D	DA0D			DB7D	DB21R	A05R	A09R			
25	RSD	DAPD			DB8D	DB22D	△	△			
27		DAPR	AIOR		DB8R	DB22R	A06R	A10R			
29				AV0D							
31				BSYD							
33	IORD	EDD		HPSD	DB9D	DB23D	△	△			
34		EDR	ASCR		DB9R	DB23R	A07R	A11R			
35	FSLD	PCD		HPID	DB10D	DB24D	△	CC3D			
36		PCR	FSR		DB10R	DB24R	A12R				
37	FSTD	DORD			DB11D	DB25D	△	CC4D			
38		DORR			DB11R	DB25R	A13R				
39	ICD	SCD			DB12D	DB26D	△	△			
40		SCR			DB12R	DB26R	A14R	RETR			
42					DB13R	DB27R	A15R	IMCR			
45					DB13D	DB27R	△	△			

△ = Must be Grounded

APPENDIX D

REFERENCE SPECIFICATIONS

123382	Design Specification, Eight-Bit Data Path Interface Subcontroller Description
127039	Master Drawing List, Subcontroller-Controller Interface
123651	Product Design Specification, Input/Output Processor, Multiplexing

APPENDIX E

GLOSSARY OF IOP/DEVICE CONTROLLER SIGNALS AND TERMS

In the following glossary, the left-hand column designates the general area of usage. The key for this column is:

SUB = subcontroller logic term
DC = controller logic term (based on design tips in this manual)
SYS = general descriptive term for a SIGMA system
RWD = RWD interface logic term

SUB	FRiD	Function response lines (i = 0-7)
SUB	DAiD	Data line drivers (i = 0-7)
SUB	DAiR	Data line receivers (i = 0-7)
SUB	DAPD	Data parity driver
SUB	DAPR	Data parity receiver
SUB	EDD	End-data driver
SUB	EDR	End-data receiver
SUB	ESR	End-service receiver
SUB	PCD	Parity check driver (requests parity checking when true)
SUB	DORD	Data/order request driver (order when true); also carries NCC1 for instruction response
SUB	IORD	Input/output request driver (output when true); also carries NCC2 for instruction response
SUB	SCD	Service call driver
SUB	SCR	Service call receiver
SUB	ICD	Interrupt call driver
SUB	FSLD	Function strobe acknowledge leading driver
SUB	FSTD	Function strobe acknowledge trailing driver
SUB	RSD	Request strobe driver
SUB	CL1R	One-megacycle clock receiver
SUB	RSTR	I/O reset receiver

SUB	RSAR	Request strobe acknowledge receiver
SUB	SIOR	SIO function indicator receiver
SUB	HIOR	HIO function indicator receiver
SUB	TIOR	TIO function indicator receiver
SUB	AIOR	AIO function indicator receiver
SUB	TDVR	TDV function indicator receiver
SUB	ASCR	ASC function indicator receiver
SUB	FSR	Function strobe receiver
SUB	HPSR	High priority service receiver
SUB	HPIR	High priority interrupt receiver
SUB	HPSD	High priority service driver
SUB	HPID	High priority interrupt driver
SUB	AVIR	Available input receiver
SUB	AVOD	Available output driver
SUB	BSYD	Busy driver
SUB	BSYR	Busy receiver
SUB	AIOC	Period during which DC presents interrupt status to subcontroller
SUB	CIH	Call interrupt high signal from DC to subcontroller
SUB	CIL	Call interrupt low signal from DC to subcontroller
SUB	CSH	Call service high signal from DC to subcontroller
SUB	CSL	Call service low signal from DC to subcontroller
SUB	DCA	DC address recognized (automatic comparison with switches)
SUB	FSC	Service connect flip-flop
SUB	FSD	Function strobe delayed (DC must return this to subcontroller when FSR occurs)
SUB	INI	Initialize after power failure
SUB	INC	Inhibit service calls during power failure
SUB	SDVii	Response data to TDV (ii = 00-07)
SUB	SSHii	Response data to SIO, HIO, TIO (ii = 00-07)
SUB	SWAi	Switch settings (i = 0-7)
SUB	TSH	(TIOR + SIOR + HIOR) · DCA

SUB	TTSH	(TIOR + SIOR + HIOR + TDVR)
SUB	LSL	Latch service low
SUB	LSH	Latch service high
SUB	LIL	Latch interrupt low
SUB	LIH	Latch interrupt high
SUB	BSYC	Priority determination signal (This DC is driving BSYD)
DC	CHEND	Channel end storage
DC	OIN	Order in
DC	OOUT	Order out
DC	DIN	Data in
DC	DOUT	Data out
DC	TO '	Terminal order
DC	START	Start control flip-flop
DC	UEND	Unusual end storage
DC	ULEG	Unusual length storage
DC	WANTS	Want service storage
DC	WANTI	Want interrupt storage
DC	BADO	Incorrect order storage
DC	N	Precedes any logic signal to denote its negation
DC	C/	Precedes any logic signal name for flip-flop to denote its clocking input
DC	S/	Precedes any logic signal name for flip-flop to denote its set conditioning input
DC	R/	Precedes any logic signal name for flip-flop to denote its reset conditioning input
DC	M/	Precedes any logic signal name for flip-flop to denote its mark (dc set) input
DC	E/	Precedes any logic signal name for flip-flop to denote its erase (dc reset) input
SYS	IOP	Input/Output Processor
SYS	M/IOP	Multiplexor IOP
SYS	S/IOP	Selector IOP

SYS	S'/IOP	Piggyback Selector IOP (attached to S/IOP)
SYS	DC	Device Controller (or data chaining - flag)
SYS	D	Device
SYS	PET	(or PET panel) DC and D tester built by Peripheral Equipment
SYS	IDC	Interface disconnect circuit (part of subcontroller)
SYS	IZC	Interrupt on zero count flag
SYS	CC	Command chaining flag
SYS	CCi	Condition code bits (i = 1-4)
SYS	ICE	Interrupt on channel end (flag)
SYS	HTE	Halt on transmission error (flag)
SYS	IUE	Interrupt on unusual end (flag)
SYS	SIL .	Suppress incorrect length (flag)
SYS	S	Skip (flag)
SYS	TIC	Transfer in channel (order)
RWD	DBiD	Data line drivers (i = 00-31)
RWD	DBiR	Data line receivers (i = 00-31)
RWD	AiiR	Address receivers (i = 00-15)
RWD	RFSR	Function strobe receiver
RWD	RFSAD	Function strobe acknowledge driver
RWD	RWDR	Read/write indicator receiver (write when true)
RWD	CC3D	Condition code 3 driver
RWD	CC4D	Condition code 4 driver
RWD	RETR	I/O reset receiver
RWD	1MCR	One-megacycle clock receiver

APPENDIX F

SUBCONTROLLER MODULE LOCATIONS

Connector Location	Module Type	Cable Designation
23	LT25	
24	LT26	(Switch comparator)
25	----	(Reserved; AT17 takes two slots)
26	AT17	J4
27	LT24	
28	AT10	J3
29	LT23	
30	AT11	J2
31	LT22	
32	AT12	J1

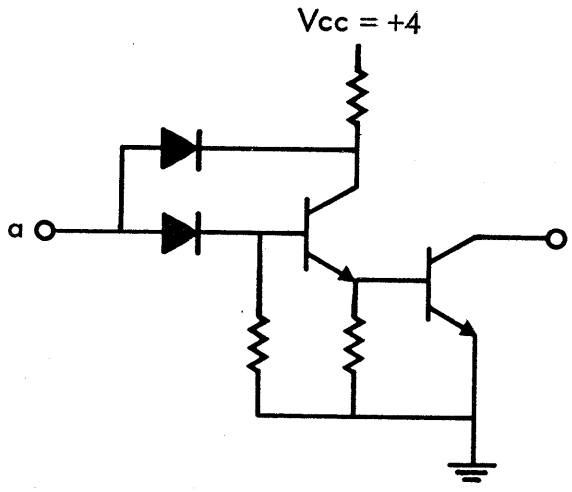
APPENDIX G

T-SERIES CIRCUITS, SPECIFICATIONS AND SCHEMATICS

Microcircuit Inverter (SDS 305), Part Number 111501

Housing:	Four circuits per TO-5 can
Turn-On Delay:	20 nanoseconds maximum (50% points)
Turn-Off Delay:	30 nanoseconds maximum (50% points)
Unloaded Output Rise & Fall Times:	20 nanoseconds maximum (10% to 90%)
Typical Operation:	18 nanoseconds average delay, 5 nanoseconds minimum (not guaranteed)
Loading:	16 gate loads maximum

Circuit Schematic:

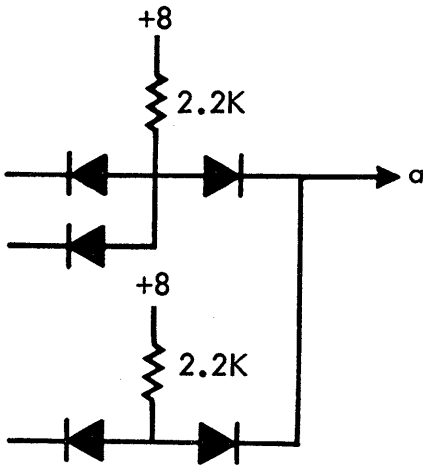


Microcircuit Inverter (SDS 305), Part Number 111501 (Continued)

Typical Required Input Gating:

Resistors: 2%

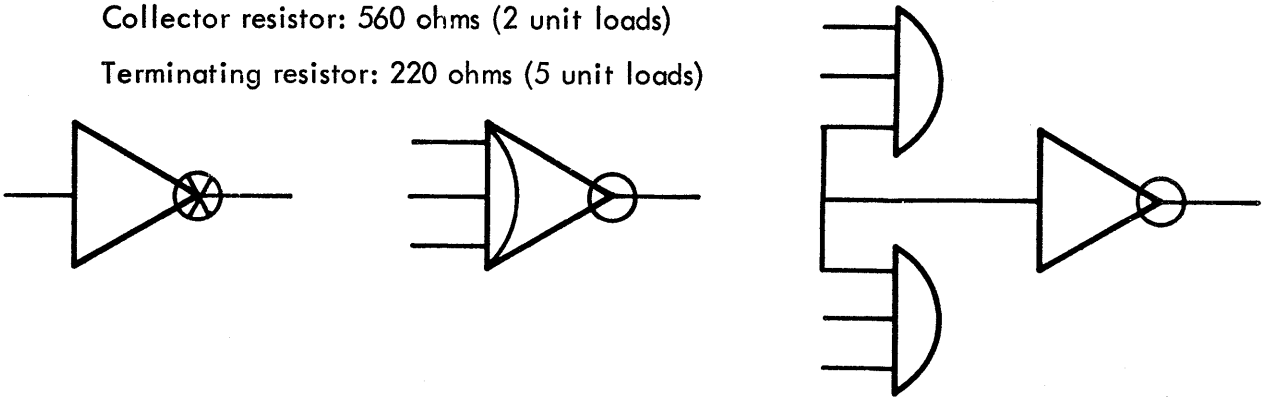
Diodes: 1N914A or equivalent



Logic Diagrams:

Collector resistor: 560 ohms (2 unit loads)

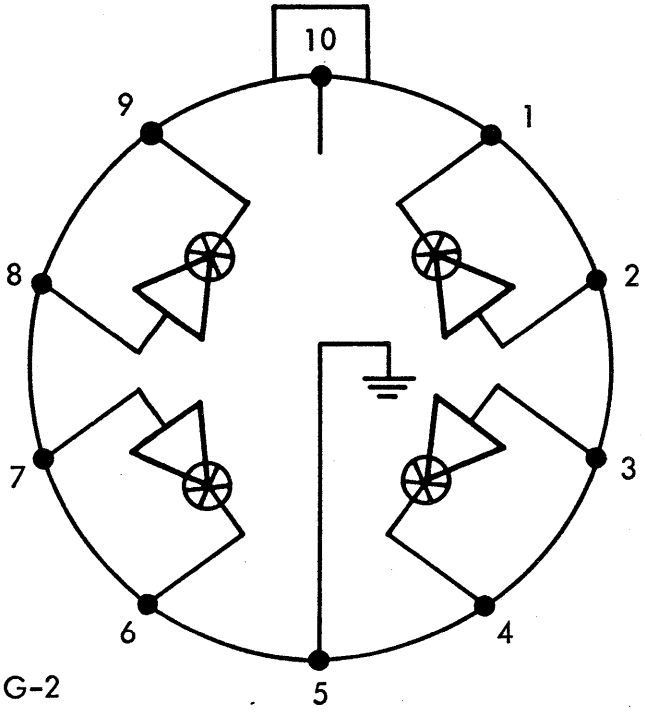
Terminating resistor: 220 ohms (5 unit loads)



(no gates, no collector resistor)

(gates and collector resistor)

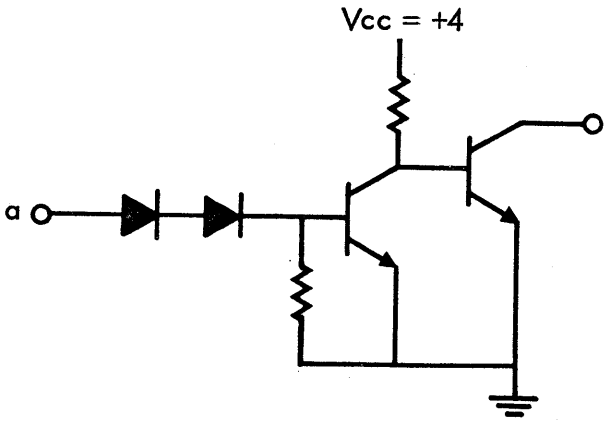
TO-5 Can Connections:



Microcircuit Buffer (SDS 306), Part Number 111502

Housing:	Four circuits per TO-5 can
Turn-On Delay:	25 nanoseconds maximum (50% points)
Turn-Off Delay:	25 nanoseconds maximum (50% points)
Unloaded Output Rise & Fall Times:	20 nanoseconds maximum (10% to 90%)
Typical Operation:	18 nanoseconds average delay, 5 nanoseconds minimum (not guaranteed)
Loading:	16 gate loads maximum

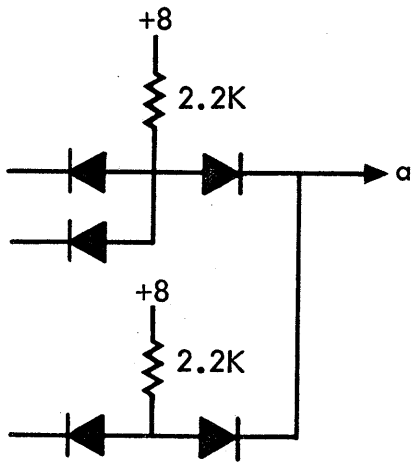
Circuit Schematic:



Typical Required Input Gating:

Resistors: 2%

Diodes: 1N914A or equivalent

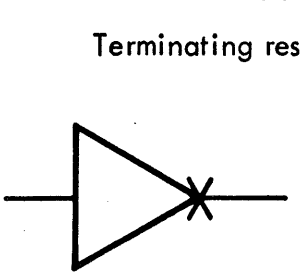


Microcircuit Buffer (SDS 306), Part Number 111502 (Continued)

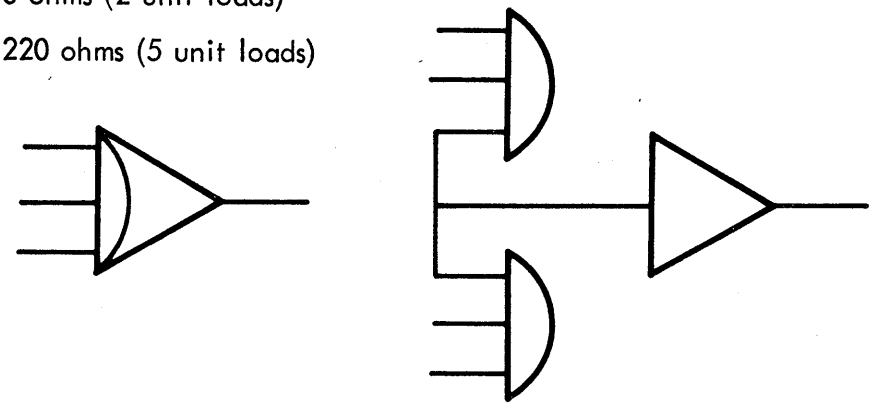
Logic Diagrams:

Collector resistor: 560 ohms (2 unit loads)

Terminating resistor: 220 ohms (5 unit loads)

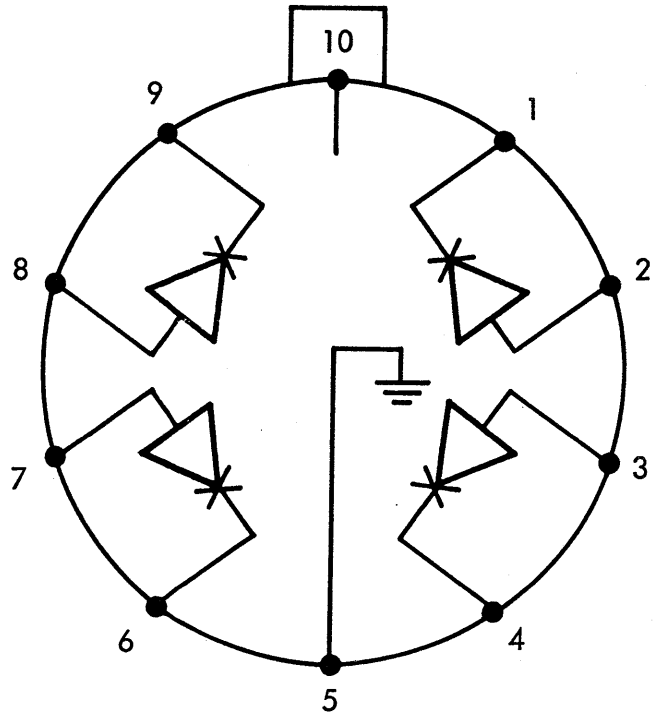


(no gates, no collector resistor)



(gates and collector resistor)

TO-5 Can Connections:



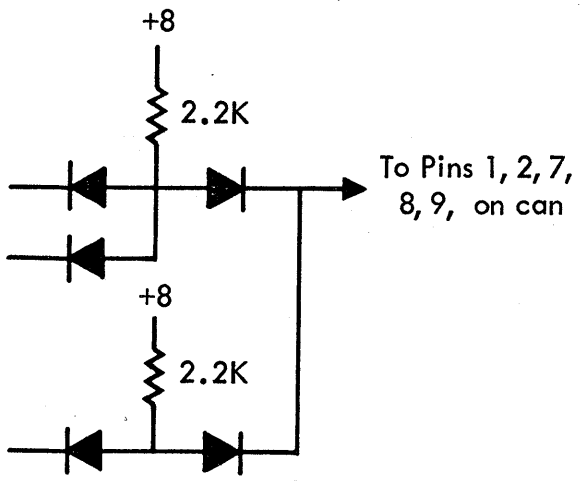
Microcircuit Flip-Flop (SDS 307), Part Number 111503

Housing:	One circuit per TO-5 can
Type:	Trailing edge trigger, with dc mark and erase
Clock Requirements:	30 nanoseconds minimum on-time (50% points) 60 nanoseconds minimum off-time (50% points)
Set/Reset Inputs:	Must be maintained at correct state from 30 nanoseconds (minimum) before clock trailing edge to 5 nanoseconds (minimum) after clock trailing edge; flip-flop completely insensitive to these inputs except in this interval
Outputs:	15 nanoseconds minimum after clock trailing edge (50% points) 70 nanoseconds maximum after clock trailing edge (50% points) 70 nanoseconds maximum after mark/erase leading edge (50% points)
Mark/Erase Inputs:	40 nanoseconds minimum on mark input to set flip-flop 40 nanoseconds minimum on erase input to reset flip-flop
Outputs:	Completely buffered (cannot alter internal flip-flop state by output collector grounding)
Loading:	16 gate loads maximum from set output, 16 from reset output
Circuit Schematic:	(refer to specification) comprises 28 transistors, 36 resistors, and 10 diodes

Typical Required Input Gating:

Resistors: 2%

Diodes: 1N914A or equivalent

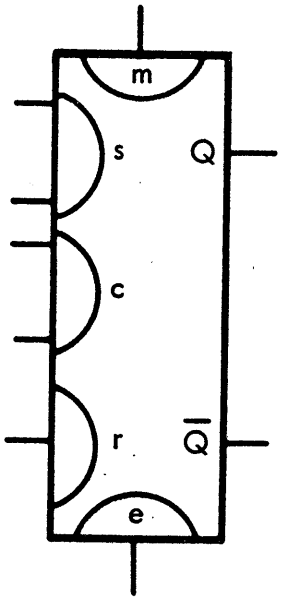


Microcircuit Flip-Flop (SDS 307), Part Number 111503 (Continued)

Logic Diagram (shown with collector resistors on set and reset outputs and typical diode gating inputs attached):

Collector resistor: 560 ohms (2 unit loads)

Terminating resistor: 220 ohms (5 unit loads)



TO-5 Can Connections: