**ComputerAutomation**

NAKED MINI® Division

18651 Von Karman, Irvine, California 92713 Tel 714 833 8830 TWX 910 595 1767

CAI Limited
Hertford House, Denham Way, Rickmansworth, Herts WD3 2XD
TEL RICKMANSWORTH 71211 • TELEX 922654

# DISTRIBUTED I/O SYSTEM

# USER'S MANUAL

91-53629-00B5 May 1977



PRINTED IN THE U.S.A.

REVISION HISTORY

| Revision | Issue Date | Comments |
|---|---|---|
| A0 | April 1975 | Original issue |
| A1 | April 1976 | Adds Positive-true General Purpose Intelligent Cable and User Microprogrammed Intelligent Cable |
| A2 | June 1976 | Adds Magnetic Tape Intelligent Cable |
| B0 | July 1976 | General revision to add DMA I/O Distributor |
| B1 | August 1976 | Revises Magnetic Tape Intelligent Cable Programming Example |
| B2 | September 1976 | Adds IEEE Intelligent Cable |
| B3 | March 1977 | Adds Dataproducts Line Printer Intelligent Cable |
| B4 | April 1977 | Miscellaneous Minor Changes |
| B5 | May 1977 | Adds 32-Bit G-P Intelligent Cable |

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

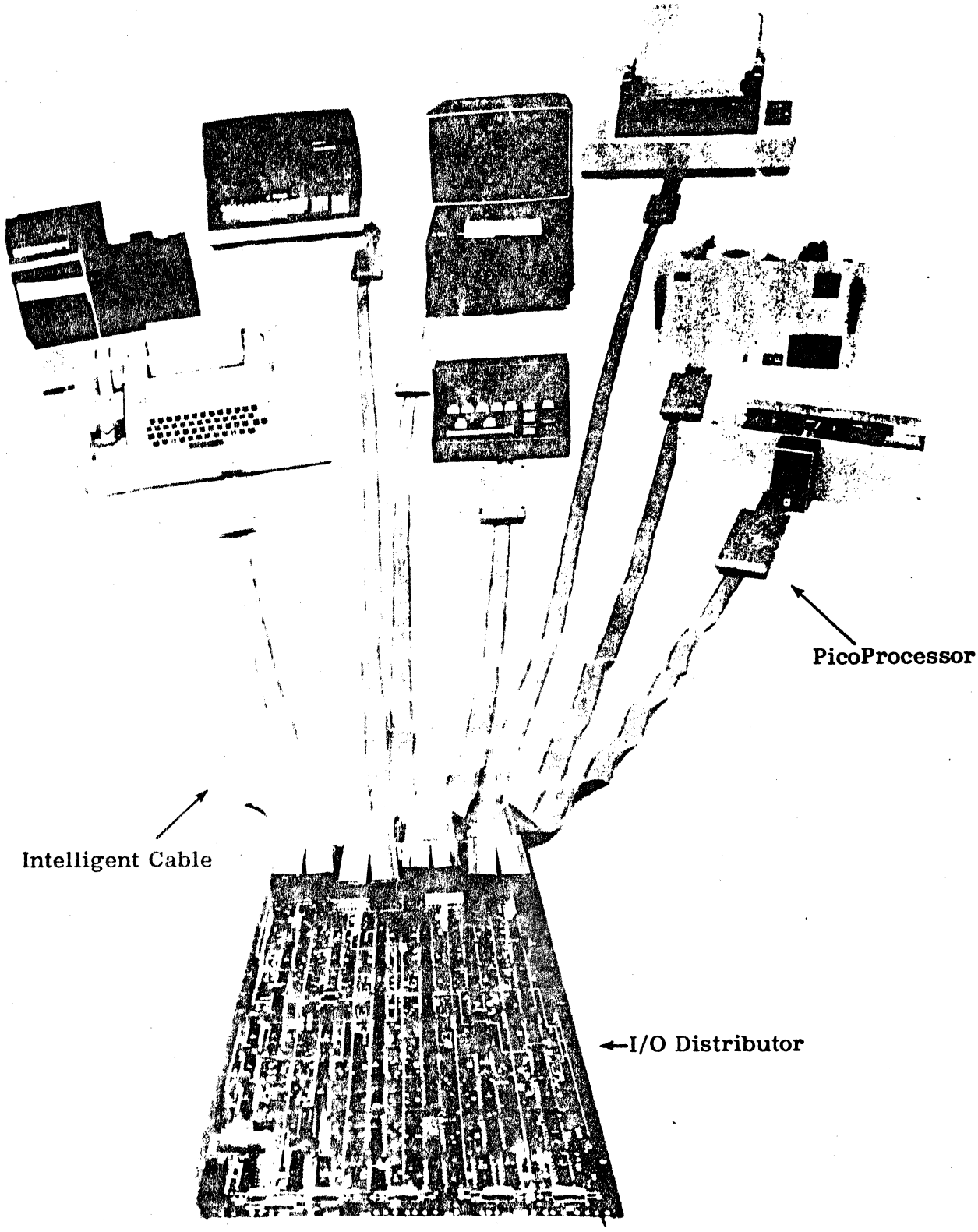
TABLE OF CONTENTS (Cont'd)

# LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont'd)

LIST OF ILLUSTRATIONS (Cont'd)

# LIST OF TABLES




Distributed I/O System

# SECTION 1

## GENERAL DESCRIPTION

The Distributed Input/Output (I/O) System provides a multiport interface control system for the LSI Family of computers. The Distributed I/O System is programmed with standard instructions, and when operating with either Auto I/O or Direct Memory Access (DMA), allows the mainline program to continue executing while data is transferred at the rate required by the peripheral. The Distributed I/O System allows attachment of any mix of Intelligent Cables.

The hardware used to implement the Distributed I/O System consists of two main parts, the I/O Distributors and the Intelligent Cables. Two I/O Distributors are available. The first is designed to handle data transfers using interrupts and the Auto I/O feature of the LSI Family. The second uses Direct Memory Access to handle the required data transfers. Either I/O Distributor can also transfer data under direct program control. Any Intelligent Cable can be attached to either of these I/O Distributors.

This manual describes the overall operation and programming of the Distributed I/O System. Section 1 is a general description of the Distributed I/O System. The system operation and general programming requirements are described in Section 2. The Intelligent Cables are individually described in Section 3. These descriptions include the applicable special programming requirements and any special installation instructions. The overall Distributed I/O System installation instructions are provided in Section 4.

For purposes of clarity, the following names are used for the Distributed I/O System hardware:

- I/O Distributor(s) - Family or collective name used to refer to any model I/O Distributor.
- Standard I/O Distributor - Refers to the I/O Distributor without Direct Memory Access capabilities.
- DMA I/O Distributor - Refers to the I/O Distributor with Direct Memory Access capabilities.
- Intelligent Cable(s) - Family or collective name used to refer to any or all models of the Intelligent Cables. The model name (e.g., Magnetic Tape Intelligent Cable) is used to refer to a specific Intelligent Cable. The device normally interfaced by the Intelligent Cable is considered included when the name, Intelligent Cable, is used in this manual.

- Channel - The Distributed I/O System components that constitute a complete interface controller. More than one channel may operate concurrently through an I/O Distributor.

## 1.1 DISTRIBUTED I/O SYSTEM CONCEPT

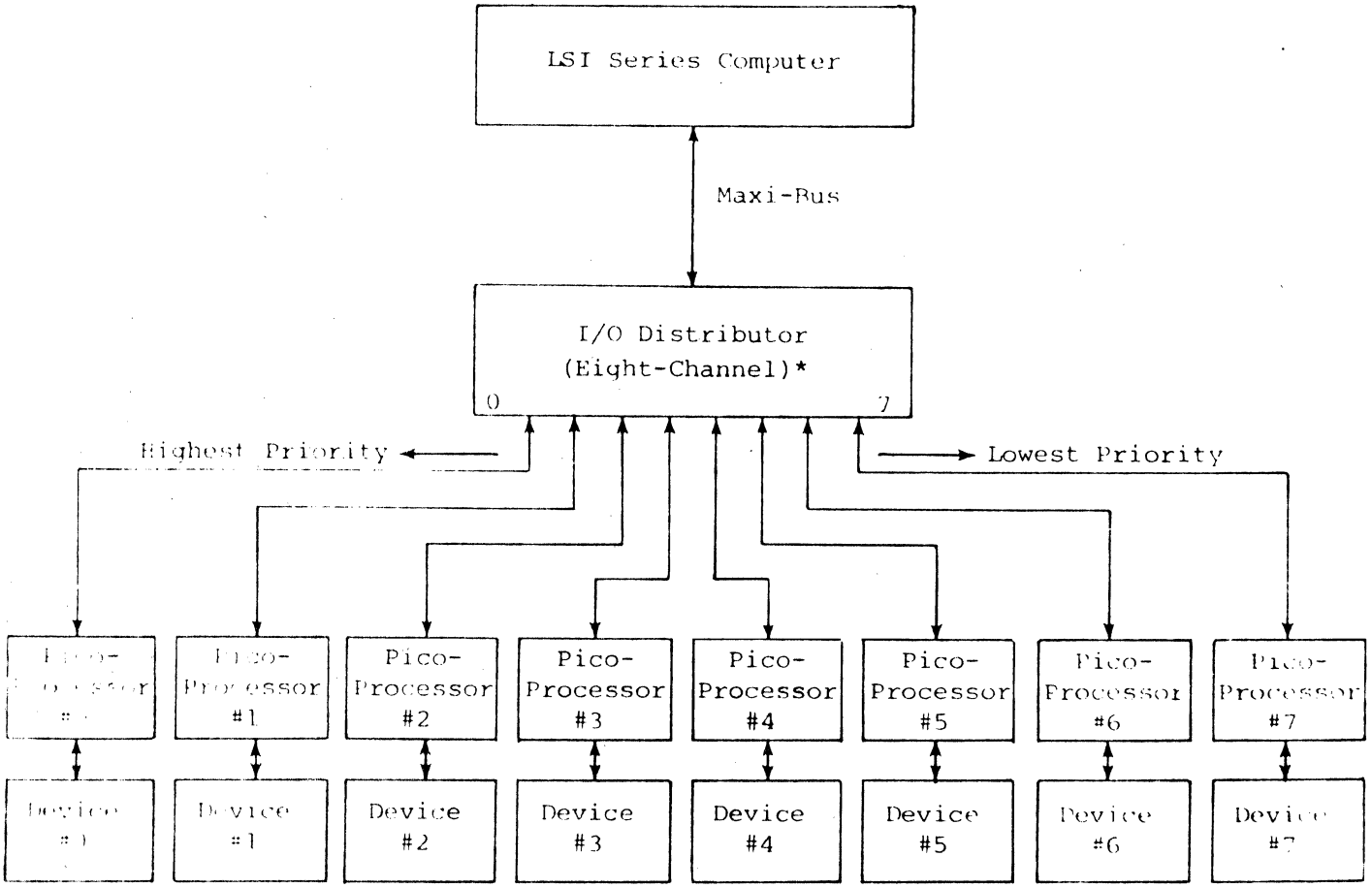
The Distributed I/O System is designed to replace the conventional single device interface controller. An I/O Distributor complete with up to four or eight Intelligent Cables can replace the same number of independent interface controllers. The physical configuration of the Distributed I/O System contributes to the packaging efficiency and the system flexibility.

The need for an interface controller usually arises when a specific peripheral must be attached to a computer. For example, the Maxi-Bus of the LSI Family of computers can interface directly with a peripheral if the peripheral observes the interface discipline of the Maxi-Bus. Very few peripheral devices, unless specifically designed to do so, will actually interface directly with the Maxi-Bus.

The interface controller resolves the interface problems by observing the Maxi-Bus interface discipline on one side, the peripherals interface discipline on the other side and internally translating one into the other.

In the Distributed I/O System, the interface controller is physically divided into two separate units. One unit, the I/O Distributor, contains the logic used to observe the Maxi-Bus interface discipline. The second unit, the Intelligent Cable, contains the logic used to observe the peripheral device interface discipline. The separation between the I/O Distributor logic and the Intelligent Cable logic exists physically, but is transparent to the computer software (see Figure 1-1).

The physical division of the interface controller logic into separate units allows the I/O Distributor logic to be used on a time-shared basis. The I/O Distributor thus expands the capabilities of an interface port to the Maxi-Bus.




* Standard 4-Channel I/O Distributors Use Channels 4 Through 7
DMA I/O Distributor Uses Either Channels 4 Through 7 or Channels 0 Through 3

Figure 1-1. Block Diagram, Distributed I/O System

## 1.2 I/O DISTRIBUTORS

The central component of the Distributed I/O System is the I/O Distributor. Constructed on a Computer Automation half-card size printed Circuit Board, the I/O Distributor can be installed in any standard chassis. The various model I/O Distributors are grouped into two operational classes: The standard models that use interrupts and the Auto I/O instruction to handle data transfers, and the DMA model that uses Direct Memory Access to handle data transfers. Either I/O Distributor can also transfer data under direct program control.

The two standard I/O Distributor models can operate with any mix of up to four (maximum) or up to eight (maximum) Intelligent Cables.

The DMA I/O Distributor is available in a single model that operates with any mix of up to four Intelligent Cables.

Each I/O Distributor develops a system clock for internal use and for use by all attached Intelligent Cables. Operating power for the Intelligent Cables is provided through the I/O Distributor. The I/O Distributor also provides strapping options for selectable baud clock rates, Parity Bit Standardization, and Special Character Detection.

The standard I/O Distributor multiplexes the Intelligent Cables to a single port on the Maxi-Bus. It assigns priorities to the data service and End-of-Block requests from the Intelligent Cables and then, in priority order, transfers these requests to the Maxi-Bus in the form of interrupts. The standard I/O Distributor is also responsible for providing the correct interrupt address when the Intelligent Cable's interrupt request is honored.

The DMA I/O Distributor includes sufficient processing capabilities to execute the equivalent of the LSI Family Auto I/O instruction. It assigns priorities to the data service requests from the Intelligent Cables and then, in priority order, makes the necessary data transfers. Since this is performed by the I/O Distributor, the mainline program is not interrupted and data transfers are made at the memory cycle rate. The End-of-Block service requests from the Intelligent Cables are assigned an interrupt priority and then, in priority order, the DMA I/O Distributor transfers these requests to the Maxi-Bus as an interrupt when there are no pending data service requests.

A micro-diagnostic is provided for test purposes as part of the DMA I/O Distributor processing capabilities. The micro-diagnostic tests the data loop for any attached Intelligent Cable. The micro-diagnostic is selected and initiated by software command from the computer. Successful completion is indicated by an interrupt to the End-of-Block subroutine.

## 1.3 INTELLIGENT CABLES - FUNCTIONAL DESCRIPTION

Peripheral devices interfaced to the computer through the Distributed I/O System are connected to an I/O Distributor by an Intelligent Cable.

Each Intelligent Cable is physically constructed as two separate cables, both connected to a PICOPROCESSOR (Figure 1-2). One cable is used to connect the I/O Distributor to the PICOPROCESSOR. . This cable is a standard flat ribbon cable. The other cable is used to connect the peripheral device to the PICOPROCESSOR. This cable is usually a short, special cable. When the Intelligent Cable is designated for use with a specific peripheral device, the special cable is supplied with the appropriate device mating connector. Detail specifications on the Intelligent Cables is provided in Section 3.

Each PICOPROCESSOR is a small, special-purpose processor designed to control the transfer of data between the peripheral device and the I/O Distributor. The PICOPROCESSOR receives both power and timing control (clock) from the I/O Distributor.

The operation of the PICOPROCESSOR is sequenced and controlled by the microprogrammed firmware instructions. The firmware instructions are arranged into one or more function programs that are tailored to the specific needs of the peripheral device.

The PICOPROCESSOR operates semi-independent of the I/O Distributor and the computer. The PICOPROCESSOR cannot initiate any of the firmware sequences. All functions are initiated by direct command from the computer software. The function initiation command takes the form of a branch command to the appropriate allowed PICOPROCESSOR firmware address. Once the function is initiated, the firmware takes control of the I/O operation, freeing the computer for other processing.

Under firmware control, the PICOPROCESSOR is able to:

- Control peripheral operation.
- Request service from the I/O Distributor.
- Modify the firmware sequence in response to detected events.

The individual descriptions of the Intelligent Cables are located in Section 3 of this manual. These include descriptions of the firmware sequences, allowed entry points to the sequences, and other special programming requirements.

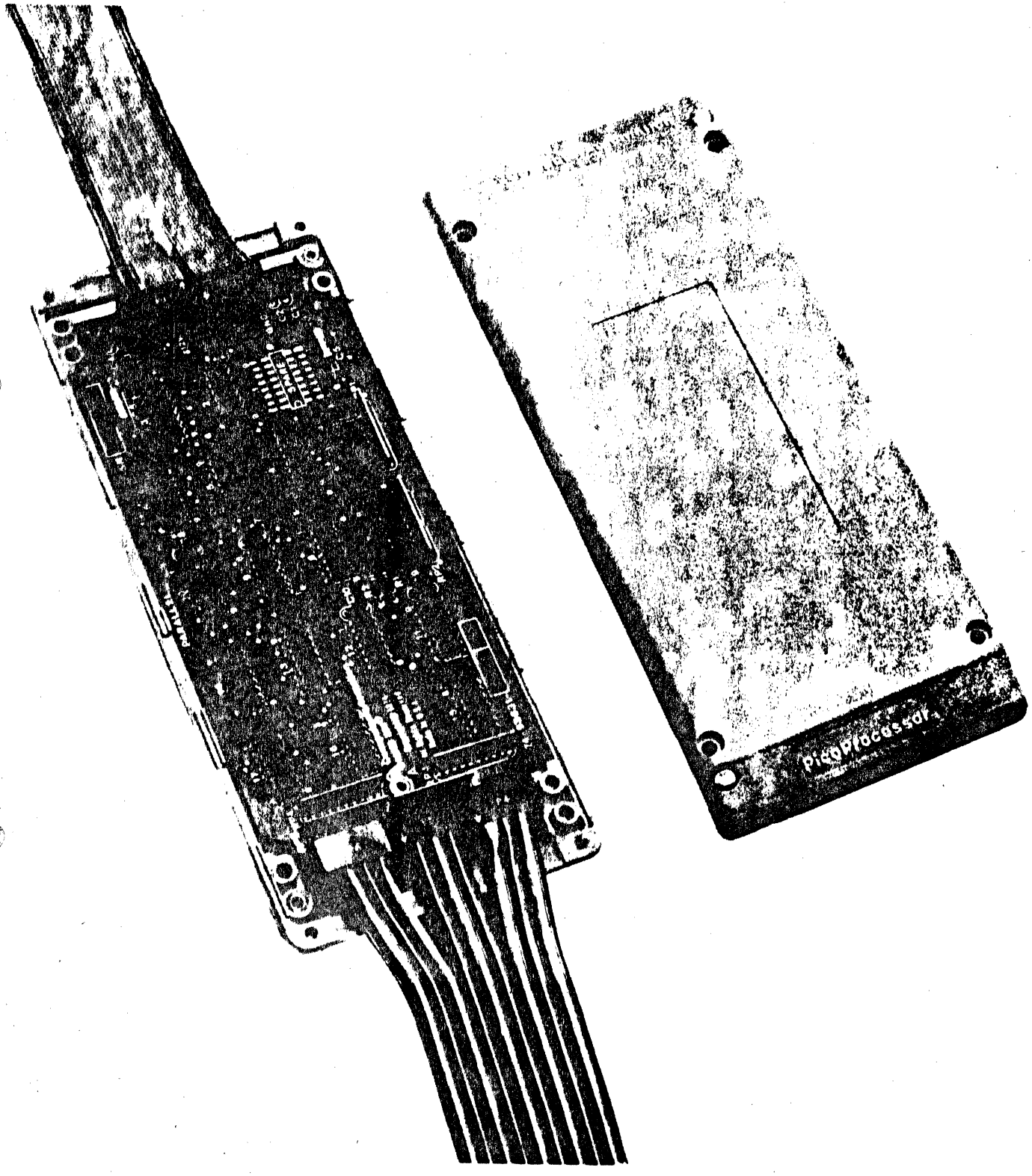



Figure 1-2. PICOPROCESSOR (Cover Removed)

## 1.4 SPECIFICATIONS

### 1.4.1 I/O Distributors Available

Standard I/O Distributors

14629-14 - 4-channel, serial or parallel data format peripherals
14629-18 - 8-channel, serial or parallel data format peripherals

DMA I/O Distributor

14674-01 - 4-channel, serial or parallel data format peripherals.

### 1.4.2 Device Data Formats

Bit serial (asynchronous and synchronous)
Parallel (8-bits output and 8- or 16-bits input)

### 1.4.3 Device Control Configurations

Simplex
Half duplex
Full duplex using two Intelligent Cables (requires two I/O Distributor channels).

### 1.4.4 Clocks

Distributed I/O System Clock

250 ns - Used by I/O Distributor and all attached Intelligent Cables.

BAUD Clocks

10 baud clock rates (16 times baud rate) are listed as follows:

| | |
|---|---|
| 19,200 | 600 |
| 9,600 | 300 |
| 4,800 | 150 |
| 2,400 | 110 |
| 1,200 | 75 |

Each channel on the DMA I/O Distributor can be set to any baud rate independent of the other channels. The 75 baud, 600 baud, and 19,200 baud rates on the standard I/O Distributor are not directly available for selection. One of these three baud rates can be included with the seven remaining baud rates as a set of eight baud rates available for selection. Each channel on the standard I/O Distributor can then be set to any of the eight available baud rates independent of the other channels.

### 1.4.5 Environmental

All components of the Distributed I/O System operate at an ambient temperature 0 to 50 degrees Celsius at 5 to 90 percent relative humidity (non-condensing).

### 1.4.6 Power Requirements

All components of the Distributed I/O System (except for the peripheral device) receive DC power from the Computer Automation LSI Family computer power supply. The maximum power requirements are as follows:

| | |
|---|---|
| I/O Distributor 14629-14 | +5V at 1.68 A |
| I/O Distributor 14629-18 | +5V at 2.18 A |
| I/O Distributor 14674-01 | +5V at 4.0 A |
| Parallel Format Intelligent Cables | +5V at 0.85 A |
| IEEE Intelligent Cable | +5V at 1.0 A |
| Serial Format Intelligent Cables | +5V at 0.70 A |
| | +12V at 0.05 A |
| | -12V at 0.065 A |

Refer to the appropriate computer handbook for the various power supply capacities. Be sure the total power demand does not exceed the rate power supply capacity.

### 1.4.7 Bandwidth

Table 1-1 contains listings of the maximum transfer rate bandwidths for various combinations of processors, memories, and I/O Distributors. The listed maximums apply to a single Intelligent Cable transferring data. All other peripherals must be in Idle, and the software in a minimum time wait loop.

Table 1-1. Maximum Bandwidth in Kilobytes

| CPU Type | I/O Distributor Type | Memory | | |
|---|---|---|---|---|
| | | Core 1600 | Core 1200 | Core 980 |
| LSI-2/60 | Standard | 60 | 80 | 86 |
| LSI-2/20 | Standard | 60 | 80 | 86 |
| LSI-2/10 | Standard | 30 | 40 | 43 |
| LSI-3/05 | Standard | 32 | 32 | 32 |
| LSI Family | DMA | 198 | 215 | 225 |

Each Intelligent Cable/peripheral has a maximum data transfer bandwidth that is device dependent. For peripheral devices that use the full handshake discipline for data transfer this bandwidth is a range from zero up to the maximum possible for the peripheral device. For peripheral devices that use the strobed discipline for data transfer this bandwidth is a specific value corresponding to the determining factor in the peripheral (e.g. magnetic tape movement speed and recording density).

Few precautions are required if only handshake discipline peripherals are used. Under worst case circumstances, the peripheral may be slowed temporarily but data is not lost. With a strobed discipline peripheral device, care must be taken to assure data is not lost.

The resulting system throughput is a function of the computer's capabilities, the number and kind of I/O Distributors, the number and kind of Intelligent Cables, and the number and kind of other peripheral controllers attached to the computer.

# SECTION 2

# OPERATION AND PROGRAMMING

This section first presents the overall description of the Distributed I/O System operation. The manner and sequence in which control is passed from one part of the system to another is included in the description. The second part of the section is a description of the various instructions and command words. Standard formats are provided along with the basic programming parameters of the Distributed I/O System.

## 2.1 SYSTEM OPERATION

The overall Distributed I/O System operation is divided among several pieces of hardware. Control of the Distributed I/O System is shifted between the system components as part of the operation sequence. The programming requirements presented later in this Section can be more easily understood with a good knowledge of the operational sequence.

The Distributed I/O System is a multi-channel system. The operational sequences of a standard I/O Distributed channel using interrupts to perform data transfers and a DMA I/O Distributor channel using Direct Memory Access to perform data transfers are described in this section. The operational sequences presented are typical of the operation of each respective channel. The transfer of data under direct program control is dependent on the software and the Intelligent Cable involved and is not described. Refer to the Intelligent Cable descriptions in Section 3.

The differences in operation between the standard and the DMA I/O Distributors are virtually transparent from a software standpoint. However, a good knowledge of the operational differences will help the programmer design software that will operate on both types of I/O Distributors with maximum efficiency.

### 2.1.1 Preparation

Prior to initiating any I/O operation with the Distributed I/O System, the computer software performs several "housekeeping" functions. The first of these is to determine the availability of the peripheral device for an I/O operation. Usually, the software checks the channel status character to make this determination. The actual content of the status character varies with each Intelligent Cable/peripheral pair and the computer software must take these differences into consideration.

If the channel is available, the computer can prepare for the I/O operation. The first part of the preparation is to load the channel's hardware assigned data service and End-of-Block interrupt locations. The information required in the data service interrupt location is the Auto I/O Op Code and channel address, the byte count for the data transfer, and the memory buffer address pointer. The information normally stored in the End-of-Block interrupt location is a jump and store instruction used to branch the mainline program to the End-of-Block subroutine.

The computer then loads a command word into either A or X register. The command word is not part of the computer's instruction set. It is a word constructed by the computer and used to control channel operation.

### 2.1.2 Operation Initiation

The initiation of a channel operation with either the standard or DMA I/O Distributor is a two-stage function. Refer to Figure 2-1, the operation initiation sequence chart for both the standard and DMA I/O Distributors.

STANDARD I/O DISTRIBUTOR

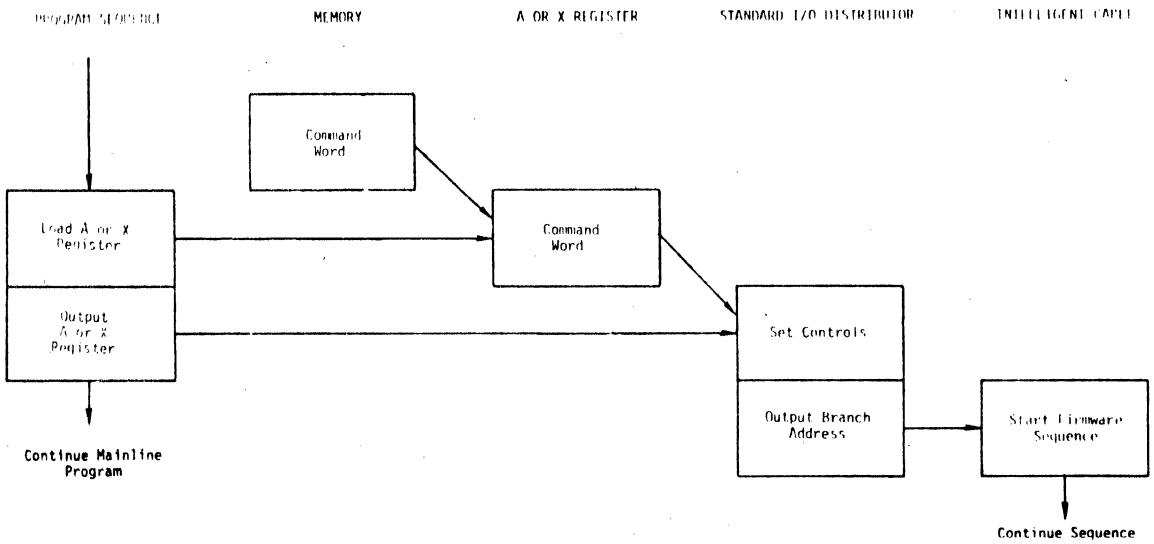
After loading the A or X register with a required command word, the computer issues the command word using the appropriate output A or X register instruction. The address and function code portion of the instruction is used to address the specific I/O Distributor and I/O Distributor channel. The function code specifies the output word is control information.

The command word contains control codes retained by the I/O Distributor for use with the addressed channel and the PICOPROCESSOR firmware branch address. This branch address is output to the addressed channel PICOPROCESSOR along with the branch (begin) control signal. The branch address supplied must be to an allowed entry point to the PICOPROCESSOR firmware.

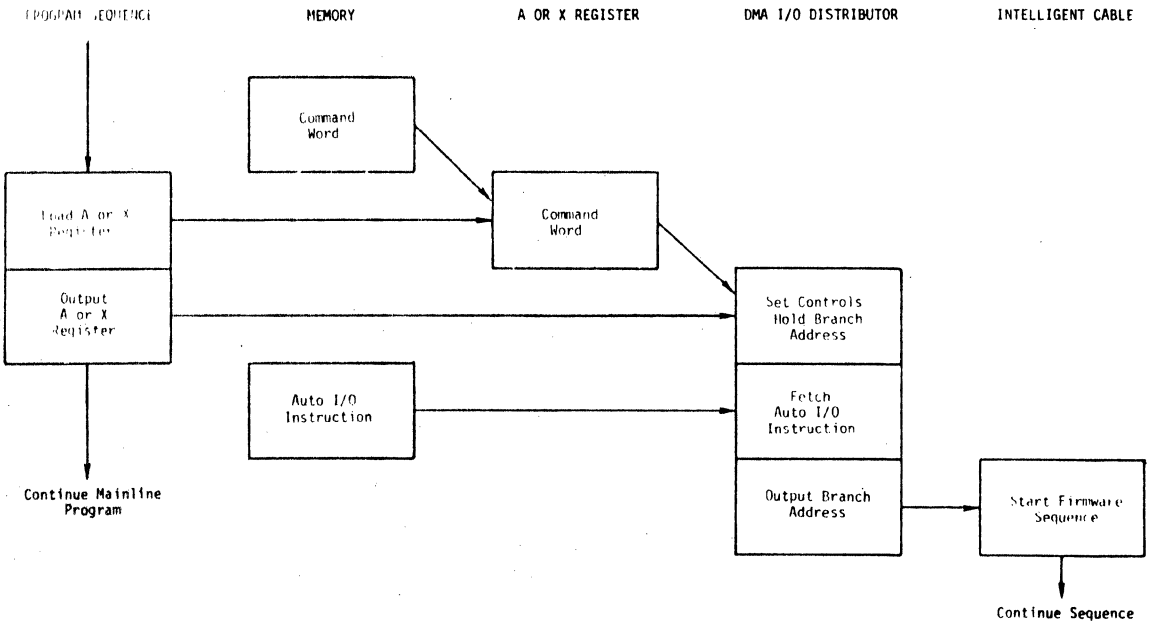
Once this PICOPROCESSOR branch command is issued, control of the I/O sequence for the channel passes to the PICOPROCESSOR. If the branch is to an I/O firmware sequence, the PICOPROCESSOR signals the peripheral that an I/O operation was requested. Depending on the type of I/O operation, the PICOPROCESSOR then waits until the peripheral is ready to receive data or has data available for transfer to the computer.

DMA I/O DISTRIBUTOR

As shown in Figure 2-1, operation initiation with the DMA I/O Distributor is similar to standard I/O Distributor operation initiation. The difference occurs when the command word is received by the DMA I/O Distributor. The control codes for the addressed channel are set as described before but the branch address is held by the DMA I/O Distributor and not issued immediately to the PICOPROCESSOR. The DMA I/O Distributor then checks the command word to see if DMA is disabled. If DMA is disabled, the DMA I/O Distributor does not fetch the Auto I/O instruction and this phase of the operation is performed similar to the standard I/O Distributor. If DMA is enabled, the DMA I/O Distributor accesses the Auto I/O instruction stored in memory at the assigned data service interrupt location for the addressed channel. The DMA I/O




Standard I/O Distributor Operation




DMA I/O Distributor Operation

Figure 2-1. Operation Initiation

Distributor stores in registers for the addressed channel, the Auto I/O Op Code, the byte count, and the memory buffer address pointer. After the fetch operation is completed, the DMA I/O Distributor then outputs the branch address to the PICOPROCESSOR and the remainder of this phase of the operation is similar to the standard I/O Distributor operation.

### 2.1.3 Data Transfers

Refer to Figure 2-2, the data transfer sequence chart for both the standard and DMA I/O Distributors.

STANDARD I/O DISTRIBUTOR

The need for a data transfer is usually sensed first by the peripheral and the appropriate signal is passed to the PICOPROCESSOR. The PICOPROCESSOR in turn sends a data service request to the I/O Distributor. When this request becomes the priority data service request to the I/O Distributor, the request is sent to the computer as an interrupt. The computer completes the instruction in progress and then recognizes the interrupt. A recognition signal is sent to the I/O Distributor. The I/O Distributor responds by providing the address of the first word of the data service interrupt area (interrupt vector). The computer then executes the channel Auto I/O instruction stored at the data service interrupt location.

The Auto I/O instruction is part of the computers instruction set (refer to the applicable computer handbook). It is a three-word instruction that has the effect of an I/O subroutine. The following operations are performed as part of the Auto I/O instruction:

The first word of the Auto I/O instruction is used to address the peripheral desired and the I/O Distributor to which the peripheral is connected. The first word also specifies the type of function.

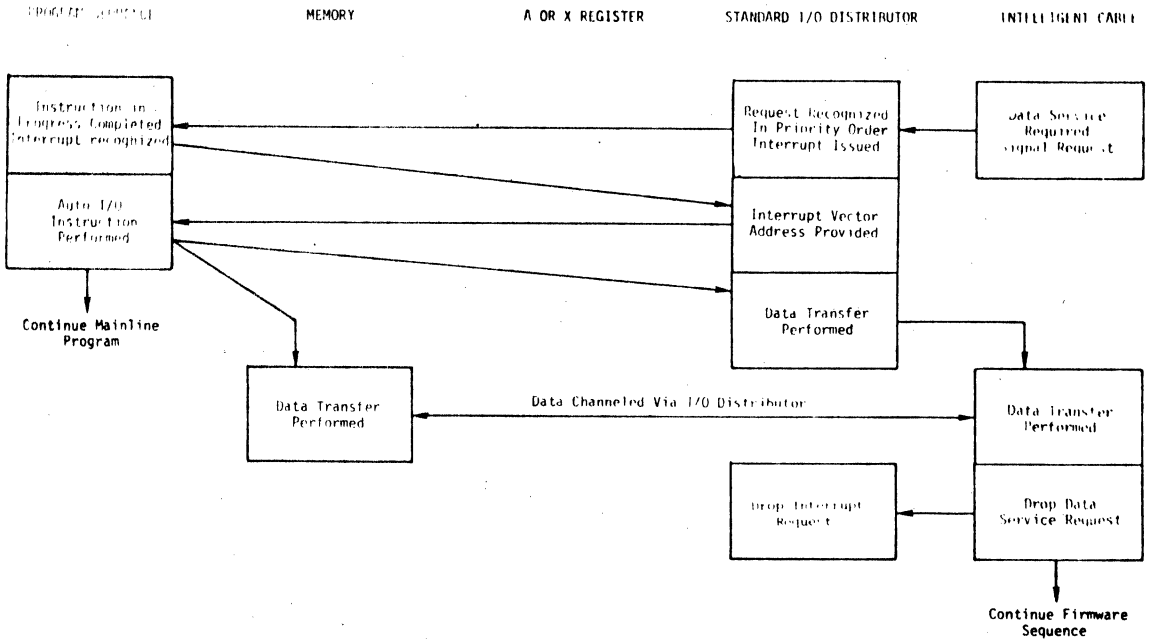
The data transfer byte count is updated and, if required, a byte count = 0 signal sent to the PICOPROCESSOR, via the I/O Distributor.

The memory buffer address pointer is updated and used to address the I/O location in memory.

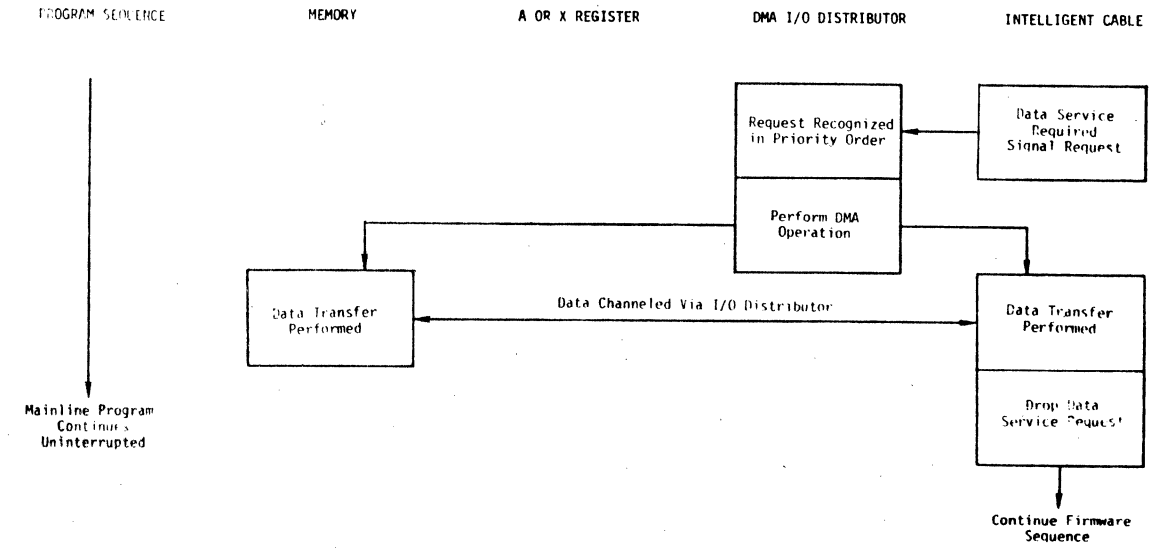
When the Auto I/O instruction is completed, the computer continues the mainline program. The computer must execute at least one mainline program instruction between interrupts. The Distributed I/O channel components drop the respective data service request and interrupt after the data transfer is completed. Additional data service requests are handled in a similar manner.

DMA I/O DISTRIBUTOR

Unlike the standard I/O Distributor data transfer sequence previously described, the DMA I/O Distributor does not interrupt the computer Mainline Program when a data transfer is required.




Standard I/O Distributor Operation




DMA I/O Distributor Operation

Figure 2-2. Data Transfer

As with the standard I/O Distributor, the need for a data transfer is usually sensed by the peripheral and is passed by the PICOPROCESSOR to the DMA I/O Distributor as a data service request. When this request becomes the priority data service request to the DMA I/O Distributor, the I/O Distributor proceeds to service the request. The following operations are performed by the DMA I/O Distributor:

- The Auto I/O Op Code held by the I/O Distributor specifies the type of data transfer, input or output.
- The data transfer byte count held by the I/O Distributor is updated and a byte count = 0 signal sent to the PICOPROCESSOR if required.
- The memory address pointer held by the I/O Distributor is updated and used to address the I/O location in memory.

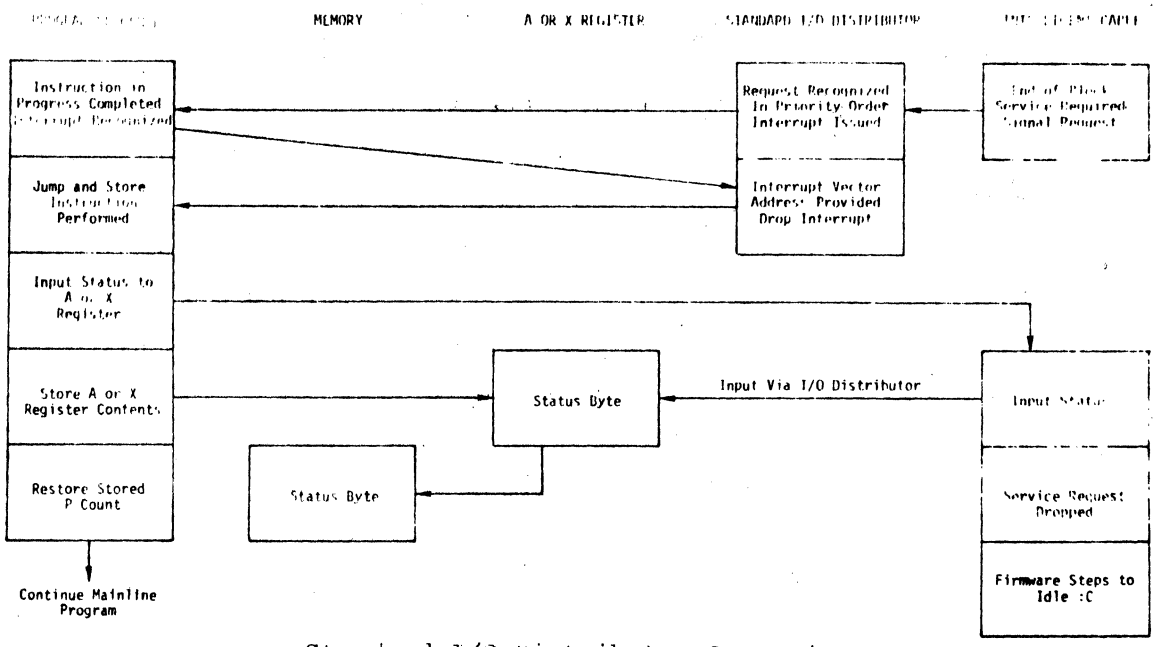
The PICOPROCESSOR drops the data service request after the data transfer is completed. Additional data service requests are handled in a similar manner.

### 2.1.4 End-of-Block Sequence

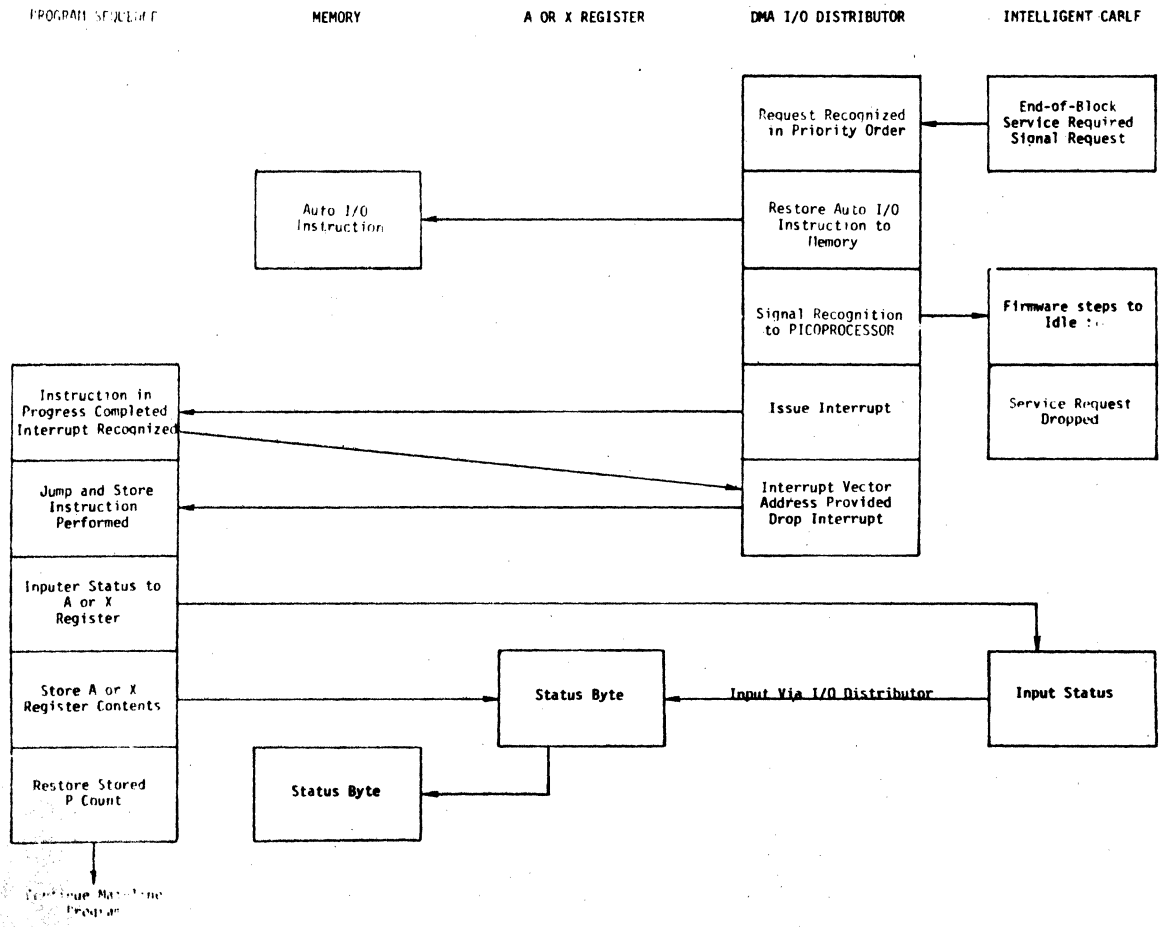
Following the data transfer when the byte count goes to zero or when other ending conditions are detected the I/O Distributor signals the PICOPROCESSOR to inhibit any further data service requests. No further response to PICOPROCESSOR operation is required until the PICOPROCESSOR detects ending conditions and the firmware steps to an instruction that sends and holds the End-of-Block service request to the I/O Distributor. Refer to Figure 2-3, the End-of-Block sequence chart for both the standard and DMA I/O Distributors.

STANDARD I/O DISTRIBUTOR

The End-of-Block service request is considered a lower priority request than a data service request. When the End-of-Block service request does become the priority request to the I/O Distributor, the request is sent to the computer as an interrupt. The computer completes the instruction in progress and then recognizes the interrupt by sending a recognition signal to the I/O Distributor. The I/O Distributor responds by providing the interrupt vector address for the first word of the End-of-Block interrupt area. The computer then executes the instruction stored at the End-of-Block interrupt location. The computer performs only one interrupt vectored instruction before continuing with the P counter sequence so the instruction normally stored at the End-of-Block interrupt location is a Jump and Store instruction. This instruction stores, for future use, the interrupted P count and jumps to the address stored in the second word at the End-of-Block interrupt location. The recognition of the interrupt by the computer is acknowledged by the I/O Distributor and the PICOPROCESSOR respectively dropping the interrupt and the End-of-Block service request. The PICOPROCESSOR firmware steps to idle completing the PICOPROCESSOR controlled I/O sequence. This passes full control back to the computer. At the same time, the computer is continuing the End-of-Block subroutine.




Standard I/O Distributor Operation




DMA I/O Distributor Operation

Figure 2-3. End-of-Block Service

The actual instructions involved for the remainder of the End-of-Block subroutine depend on the Intelligent Cable involved and specific software needs. Normally, status is requested as part of the End-of-Block subroutine for a data transfer operation. The input A or X instruction is used for this operation. The address and function portion of the instruction addresses the I/O Distributor and the I/O Distributor channel. The function code specifies the input word is an 8-bit channel status byte. The input A or X instruction causes the PICOPROCESSOR to input the contents of the status register to A or X register. The computer can then examine or store for use later, the contents of A or X register.

The final instruction in the End-of-Block subroutine normally restores the interrupted P count to the P counter. The mainline program picks up processing at the instruction following the interrupt.

DMA I/O DISTRIBUTOR

When an End-of-Block service request becomes the priority request, the DMA I/O Distributor first determines whether the channel operation was performed with Direct Memory Access disabled. If DMA was disabled, the End-of-Block service request is handled the same as with the standard I/O Distributor.

If Direct Memory Access was enabled during the channel function, the DMA Distributor does not immediately issue the interrupt. Prior to issuing the interrupt, the DMA I/O Distributor performs some additional housekeeping.

The DMA I/O Distributor first restores the final byte count and final memory buffer address pointer to the corresponding locations within the channel data service interrupt location. The DMA I/O Distributor then signals acceptance of the End-of-Block service request to the PICOPROCESSOR. This allows the firmware to step to idle and drop the service request.

The DMA I/O Distributor then issues the interrupt and the remainder of the operation is the same as with the standard I/O Distributor.

## 2.2 PROGRAMMING

The Distributed I/O System is programmed using standard LSI Family instructions. The instructions used for communication with the Distributed I/O System and for programmed I/O are: Input to A Register (INA), Input to X Register (INX), Output A Register (OTA), and Output X Register (OTX). The automatic I/O instructions used by the Distributed I/O System are: Automatic Input Byte (AIB) and Automatic Output Byte (AOB). The specific use of these instructions in conjunction with the Distributed I/O System is described in this section. For further information on these instructions, refer to the applicable computer reference manual.

In addition to the standard instructions, a special command word is used to directly control the operation of the PICOPROCESSOR. The use of the command word is also described in this section.

### 2.2.1 Input and Ouput Instructions

The computer uses the input and output instructions to:

- Output the command word to the addressed PICOPROCESSOR
- To request channel status from the addressed PICOPROCESSOR
- For data transfer under direct program control

These instruction words consist of three parts: an Op Code, a device address, and a function code. The format of these instruction words, as used with the I/O Distributor, is shown in Figure 2-4.

The Op Code specifies the function to be performed. In this case, the contents of A or X register is output to the addressed PICOPROCESSOR or a word of information from the addressed peripheral is entered into A or X register.

The eight least significant bits of instruction word make up the device address and the function code. The device address is further separated into the I/O Distributer address (Bits 4 through 7) and the channel address (Bits 1 through 3).

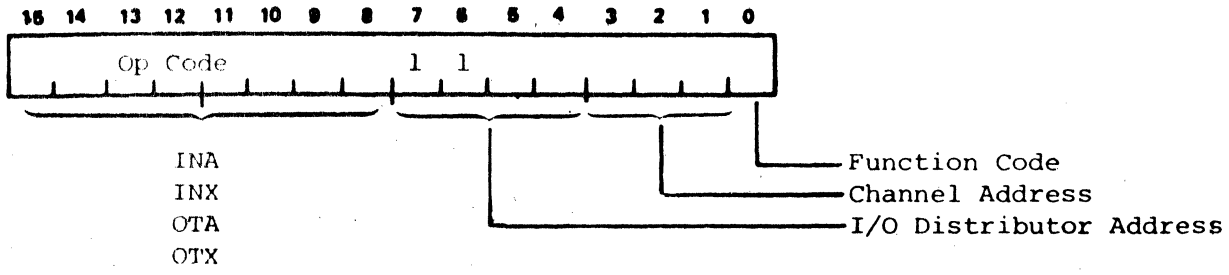



Figure 2-4. I/O Instruction Format

The I/O Distributor requires that Bits 6 and 7 must be "1" to select any I/O Distributor. This restricts the number of I/O Distributor selections to four. These are:

- I/O Distributor 0, Address :C
- I/O Distributor 1, Address :D
- I/O Distributor 2, Address :E
- I/O Distributor 3, Address :F

The I/O Distributors are shipped, factory wired, to accept :F as the I/O Distributor address. Refer the Section 4 of this manual for information on changing the address of an I/O Distributor.

The three bits of the channel address field, taken by themselves, correspond to the channel numbers of the I/O Distributor. The channel numbers are permanently assigned to the Intelligent Cable connectors on the I/O Distributor. The Intelligent Cable thus assumes the channel number of the connector when it is attached to the I/O Distributor.

The function code is the least significant bit in the instruction word and is used to select either control or data information transfers. The logical 1 function code selects the transfer of control information (command word output and channel status input). The logical 0 function code selects the transfer of data and is used only with program controlled data transfers.

For programming purposes, the channel address and the function code are taken together as a single hexadecimal character. Table 2-1 lists the hexadecimal characters used to address each of the I/O Distributor's channel numbers for both control and data transfers.

Table 2-1. Channel Address/Function Code List

| Channel Number | Control Transfers (LSB=1) | Data * Transfers (LSB=0) |
|---|---|---|
| 0 | :1 | :0 |
| 1 | :3 | :2 |
| 2 | :5 | :4 |
| 3 | :7 | :6 |
| 4 | :9 | :8 |
| 5 | :B | :A |
| 6 | :D | :C |
| 7 | :F | :E |

*Programmed I/O Only

2.2.2 Distributed I/O System Command Word

The command word is used by the computer to control the operation on a Distributed I/O System channel. The various configurations of the command word used for this purpose are not part of the computer's instruction set. The command words must be constructed individually and then output to the appropriate Distributed I/O System channel. The OTA or OTX instruction (LSB = 1) is used to output (issue) the command word.

This section includes a description of the control significance of each command word bit followed by the descriptions of the various command configurations of the command word. Not all configurations are used with each Intellgent Cable. Refer to the individual Intelligent Cable descriptions in Section 3 to determine the specific subset of command word configurations used.

COMMAND WORD FORMAT

The format of the command word is shown in Figure 2-5.

The eight most significant bits of the command word are the individual control bits and the eight least significant bits make up the various fields used in conjunction with specific control bits.

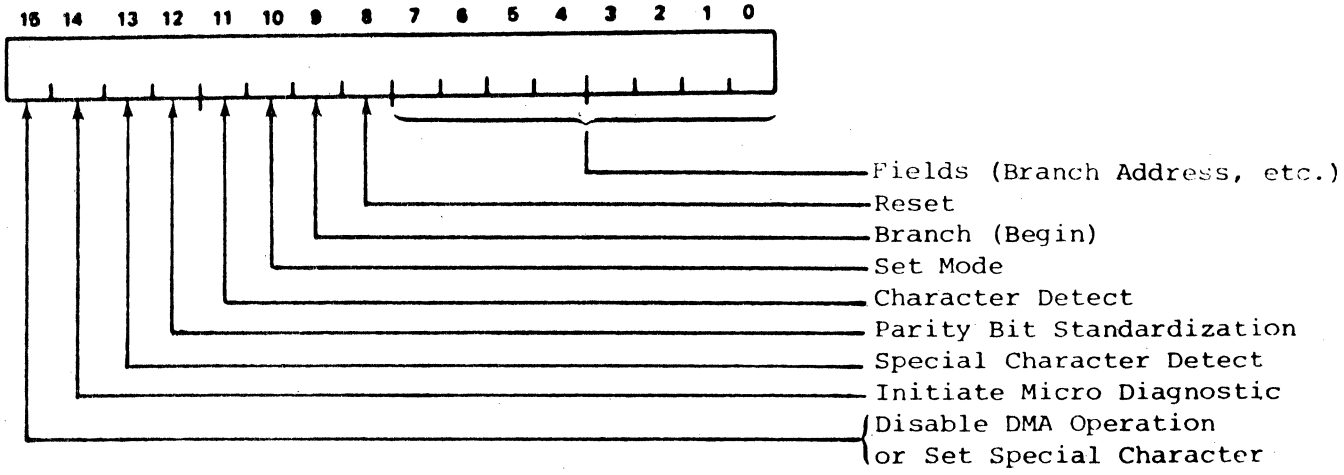



Figure 2-5. Distributed I/O System Command Word Format

MODE FIELD

The mode field is used in two different configurations, 4-bit and 8-bit. When limited to four bits (command word Bits 0 through 3), the mode field can share the least significant eight bits with the branch address field. The bits of the mode field are used to provide static control lines to the peripheral or control special features within the PICOPROCESSOR. Not all Intelligent Cables require mode bits and when used, the functions controlled vary depending on the type and operation of the Intelligent Cable. Refer to Section 3.

BRANCH ADDRESS FIELD

The branch address field, Bits 4 through 7 of the comamnd word, provides the PICO-PROCESSOR firmware entry point address. The PICOPROCESSOR firmware must be in idle, firmware address :0, before the branch address is output and the branch address must be an allowed entry point to the firmware.

SPECIAL CHARACTER FIELD (DMA I/O DISTRIBUTORS ONLY)

The special character field uses Bits 0 through 7 to load a special character into the channel character-detect register.

RESET

A logical 1 in Bit 8 of the command word initiates a reset operation on the addressed channel. In the PICOPROCESSOR, the reset initializes the PICOPROCESSOR logic and, when applicable, issues a reset to the attached device.

BRANCH (BEGIN)

A logical 1 in Bit 9 of the command word signals the PICOPROCESSOR to branch to the address provided in the branch field. At the same time, the I/O Distributor stores the configuration of command word Bits 11, 12, and 13 for use with the computer addressed Distributed I/O System channel.

SET MODE

A logical 1 in Bit 10 of the command word signals the PICOPROCESSOR to load the mode field bits into its mode register. A logical 0 allows the mode register configuration to remain unchanged.

CHARACTER DETECT

A logical 1 in bit 11 of the command word causes the I/O Distributor to scan the channel data during the subsequent input data transfer for a predetermined character. When the character is detected, the I/O Distributor issues a byte count = 0 to the channel's PICOPROCESSOR to terminate the data transfer. A logical 0 disables character detection.

NOTE

Character detection in the standard I/O Distributor can detect only the 7-bits of the ASCII carriage return character (:OD).

PARITY BIT STANDARDIZATION

A logical 1 in Bit 12 of the command word causes the I/O Distributor to replace the most significant data bit (Bit 7) of all channel input data characters during the subsequent data transfer. The replacement occurs independent of the actual received bit value. The DMA I/O Distributor can be strapped to use either a logical 1 or logical 0 as the replacement bit. The standard I/O Distributor is restricted to a logical 1 as the replacement bit. A logical 0 in Bit 12 disables parity bit standardization.

SPECIAL CHARACTER DETECT (DMA I/O Distributor Only)

A logical 1 in Bit 13 of the command word causes the DMA I/O Distributor to scan the channel data during the subsequent data transfer using the stored special character for the channel. Character detect, Bit 11, must be a logical 1 to enable any character detection. A logical 0 in Bit 13 (Bit 11 = 1) causes the DMA I/O Distributor to scan for the 7-bits of the ASCII carriage return character (:OD).

INITIATE MICRO DIAGNOSTIC (DMA I/O Distributor Only)

A logical 1 in Bit 14 of the command word causes the DMA I/O Distributor to perform the internal micro diagnostic on the addressed channel. Use of the internal micro diagnostic should be restricted and not included as part of any system programs.

DISABLE DMA OPERATION OR SET SPECIAL CHARACTER (DMA I/O Distributors Only)

A logical 1 in Bit 15 of the command word is used for two different operations. If the branch bit (Bit 9) is a logical 0, the I/O Distributor stores the configuration of the special character field (command word Bits 0 through 7) as the channel special character. If the branch bit is a logical 1, the addressed channel is restricted from DMA operation. A logical 0 in Bit 15 allows normal operation.

COMMAND WORD SET

The 18 permissible configurations of the eight most significant command word bits make up the command word set of the Distributed I/O System. The operations controlled by these 18 command words are described in this section. The full 16-bit configuration of each command word is shown along with the corresponding hexidecimal skeleton. (Hyphens in the skeleton represent field contents that must be provided).

RESET COMMAND WORD

The reset command word (Figure 2-6) is used to initialize the PICOPROCESSOR and peripheral on the addressed channel. In addition, any pending service requests are cancelled. Interrupts should be disabled when a reset command word is issued to avoid cancelling an interrupt in the process of being serviced. Since the reset command preempts any other operation in progress, it should be used only when necessary.

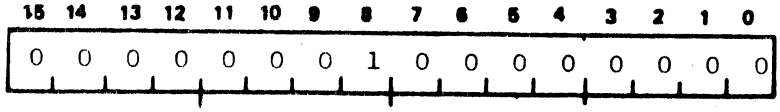



Figure 2-6. Reset Command Word

BRANCH COMMAND WORD

The branch command is used to initiate all PICOPROCESSOR firmware sequences. The configuration shown in Figure 2-7 is the basic form of the branch command word. Of the 18 permissible command words, 14 are in combination with the branch command.

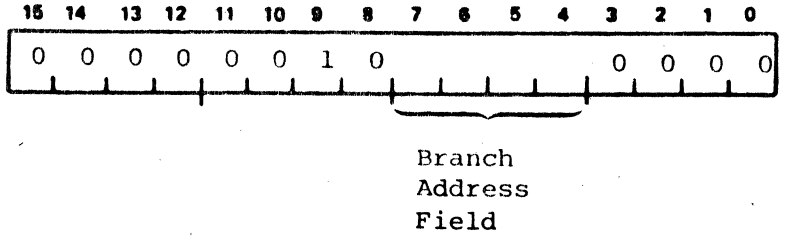



Figure 2-7. Branch Command Word

SET MODE COMMAND WORDS

The set mode commands are used to load the contents of the mode field into the PICO-PROCESSOR mode register. Figure 2-8 shows the basic forms of the set mode command word and the set mode command combined with the branch command. The form of the set mode command word used depends on the Intelligent Cable for the channel. If the Intelligent Cable's PICOPROCESSOR requires up to four mode bits, the four bit mode field must be used. If the PICOPROCESSOR requires more than four mode bits, the eight bit mode field must be used. The combining of the set mode command with any other command is restricted to those Intelligent Cables that require a maximum of four mode bits.

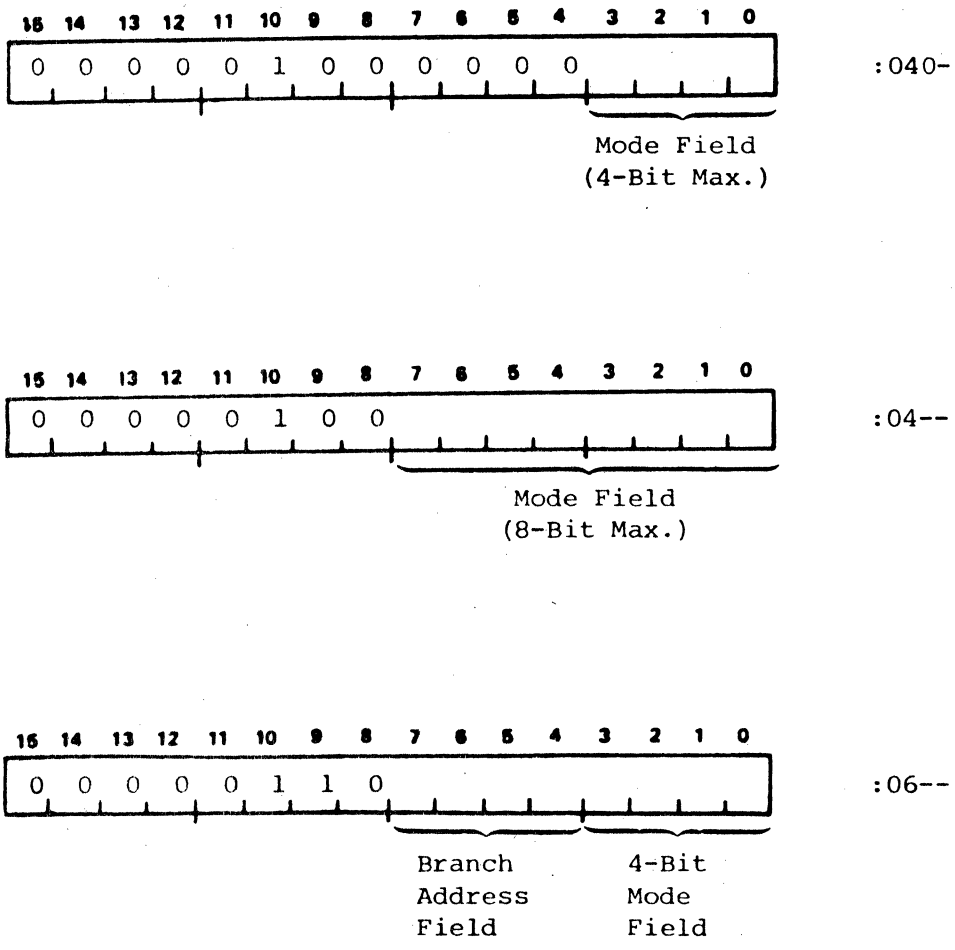
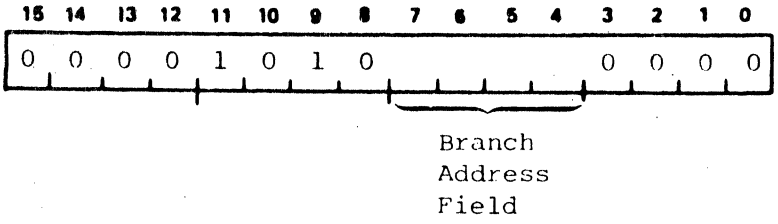
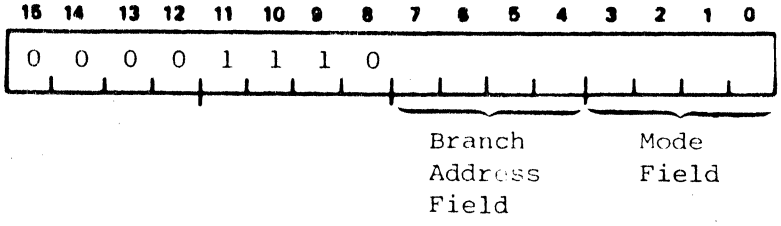



Figure 2-8. Set Mode Command Word

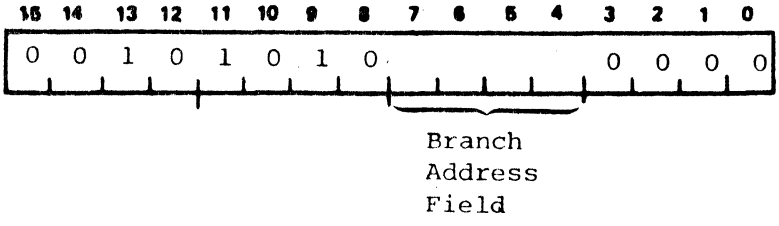



:0A-0

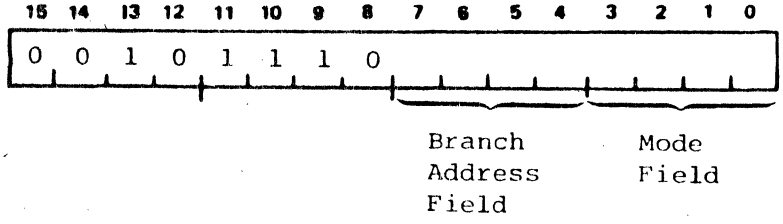



:0E--

ASCII Carriage Return Detect




:2A-0




:2E--

Special Character Detect (DMA I/O Distributor Only)

Figure 2-9. Character Detect Command Words

CHARACTER DETECT COMMAND WORDS

The character detect command words are used to enable the character detection for the addressed channel. The character detect command words (Figure 2-9) are all combined words since the branch control bit (Bit 9) is required to effect the enabling of the feature. The mode control bit (Bit 10) can also be combined with character detect. The standard I/O Distributor is limited to the detection of the ASCII carriage return character (enabled by Bit 11) while the DMA I/O Distributor can detect either the ASCII carriage return character or the stored channel special character (enabled by Bit 13).

NOTE

The DMA I/O Distributor character detection does not scan for both characters at the same time.

LOAD SPECIAL CHARACTER COMMAND WORD

The load special character command word (Figure 2-10) is used to load the configuration of the eight least significant bits into the special character register for the addressed DMA I/O Distributor channel. This command must not be combined with any other command word.

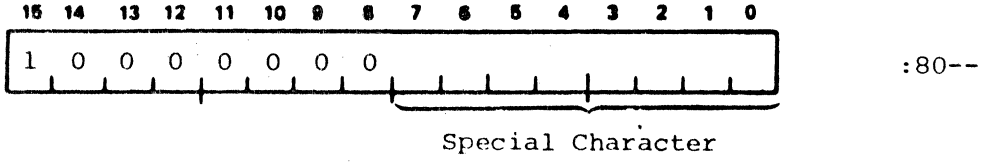
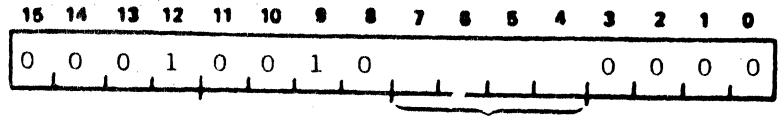
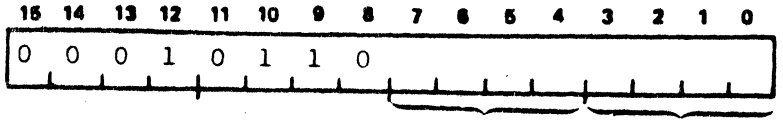



Figure 2-10. Load Special Character Command Word (DMA I/O Distributor Only)

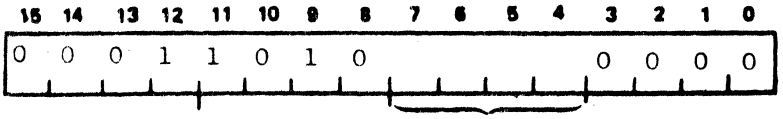



:12-0

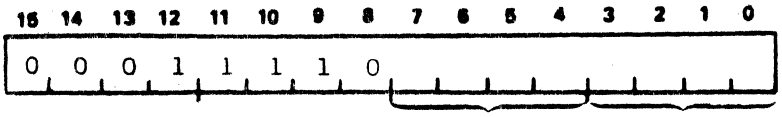



:16--

Parity Standardization Only

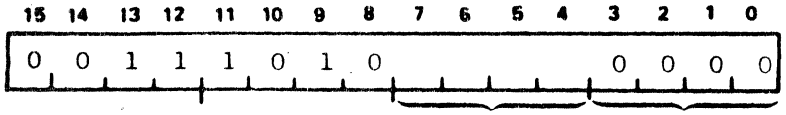



:1A-0

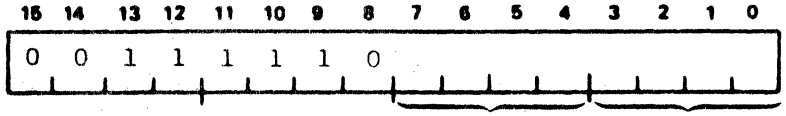



:1E--

Parity Standardization Combined with ASCII Carriage Return Detect




:3A-0




:3E--

Parity Standardization Combined with Special Character Detect
(DMA I/O Distributor Only)

Figure 2-11. Parity Standardization Command Words

PARITY STANDARDIZATION COMMAND WORDS

The parity standardization command words (Figure 2-11) are used to enable the parity standardization for the addressed channel. Bit 12 is used to enable parity standardization. The parity standardization command words are all combined words since the branch control bit is required to effect the enabling of the feature.

DISABLE DMA COMMAND WORDS

The disable DMA command words are used to inhibit the DMA operation on the addressed channel. The disable DMA command words (Figure 2-12) are all combined words since the branch control bit is required to effect the disabling of DMA. The disable DMA must not be combined with character detect or parity standardization. The disable DMA is used in the following types of operation:

- When program controlled data transfer is to be performed.
- When interrupts are to be disabled as for example, during AutoLoad.
- For program efficiency. If a data transfer will not occur (e.g. sense ready with the Magnetic Tape Intelligent Cable), the disable DMA eliminates the I/O Distributor overhead required to fetch the Auto I/O instruction from the computer memory.

To disable all DMA from an I/O Distributor, a separate disable DMA command word must be issued to each channel. Though issued separately, the disable DMA command for a channel is not totally independent. If a data service request occurs on a channel with DMA disabled, this data service request will mask any data service request on all lower priority channels, regardless of whether DMA is disabled for those channels.

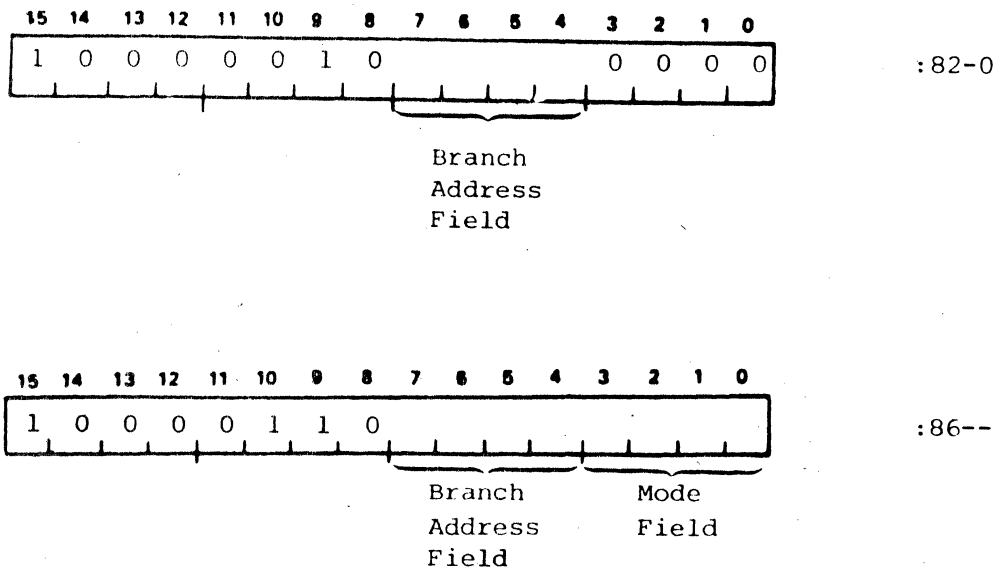



Figure 2-12. Disable DMA Command Words

INITIATE MICRO DIAGNOSTIC COMMAND WORD

The initiate micro diagnostic command word (Figure 2-13) is used to initiate the DMA I/O Distributor's micro diagnostic for the addressed channel. An End-of-Block interrupt indicates successful completion of the diagnostic. Use of this command should be restricted.

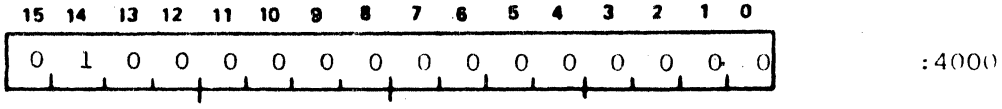



Figure 2-13. Initiate Micro Diagnostic

2.2.3. Auto I/O Instruction

The Auto I/O instruction is part of the computers standard instruction set. It is a single instruction that is three words in length. The three words that make up the Auto I/O instruction must be stored in consecutive memory locations.

The first word of the Auto I/O instruction is the instruction proper. The format of this word is shown in Figure 2-14. The format of this word of the Auto I/O instruction is the same as the format of the input and output instructions described previously. The Op Code specifies the direct transfer of a data byte between memory and the addressed I/O Distributor channel. The I/O Distributor address, channel address and function code are the same as described previously; however, the function code must be set to zero. The Auto I/O instruction is not used to transfer control information.

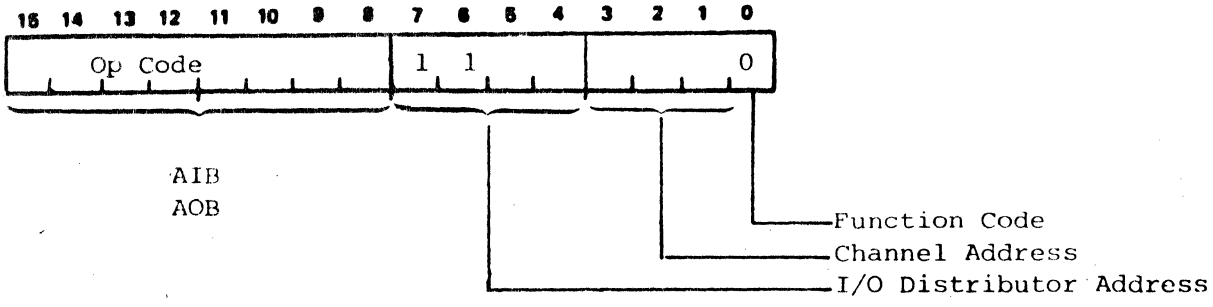



Figure 2-14. Automatic I/O Instruction Word

The second word contains the two's complement of the byte count for the data transfer. The third word contains the byte address pointer specifying one less than the first (lower-order memory) location in the memory data buffer.

The standard I/O Distributor requires the next consecutive memory location following the Auto I/O instruction be all zeros.

The standard I/O Distributor does not directly use the Auto I/O instruction but rather interrupts the computer and addresses the location of the Auto I/O instruction for the interrupting channel. The DMA I/O Distributor actually uses the Auto I/O instruction as the channel programming for the data transfer. This way the instruction setup for both the standard and DMA I/O Distributors remains the same.

### 2.2.4 Interrupt Vectoring

Both the standard and the DMA I/O Distributors generate interrupts to the computer. Once the interrupt is recognized, the I/O Distributor must then provide a memory address to the computer for use as the next instruction address. This process is call interrupt vectoring.

Each channel of the standard I/O Distributor is able to provide two different interrupt vector addresses. One address is provided whenever data service is required and a second address is provided when the End-of-Block is encountered. The DMA I/O Distributor interrupts only when the End-of-Block is encountered and provides only the corresponding address.

The actual addresses provided by the I/O Distributor are controlled by the I/O Distributor hardware. The following conventions determine the hardware assignment of the interrupt vector address:

- The interrupt vector addresses are assigned within a block of 64 consecutive memory addresses.
- Each channel is allocated eight consecutive memory addresses.
- An interrupt for data service is vectored to the channel low order memory address.
- An interrupt for End-of-Block service is vectored to the channel low order memory address plus four.
- The location of the 64 address block is selectable (in increments of 64 addresses) by strapping on the I/O Distributor.

Figure 2-15 shows all the interrupt vector addresses for the 64 address block selected by the standard (factory) I/O Distributor strapping. This block starts at memory address :C0 and goes through address :FF.

Figure 2-15 also shows the format of the eight consecutive memory addresses for channel 0. The format for the remaining channels is the same. If the I/O Distributor is strapped for a different 64 address block (refer to Section 4 of this manual for details on strapping), the vector addresses would be changed. Each vector address in the new block would have the same relationship to the new low order memory address as the relationship shown in Figure 2-15.

The Distributed I/O System requires that an Auto I/O instruction be stored in the data service area as shown. The End-of-Block subroutine is not restricted in the same manner. Normally, the form shown in Figure 2-15 is used to store the existing P count, jump to the End-of-Block subroutine, perform the subroutine, and then return to the mainline program at the stored P count address.

| Channel | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Data Service Vector Address | :C0 | :C8 | :D0 | :D8 | :E0 | :E8 | :F0 | :F8 |
| End-of-Block Vector Address | :C4 | :CC | :D4 | :DC | :E4 | :EC | :F4 | :FC |

| | | |
|---|---|---|
| :C0 | Auto I/O Op Code & Address | Data Service |
| :C1 | Byte Count | |
| :C2 | Memory Buffer Pointer | |
| :C3 | All Zeros | |
| :C4 | Jump and Store(JST * $+1) | EOB Service |
| :C5 | EOB Subroutine Address | |
| :C6 | Unused | |
| :C7 | Unused | |

Figure 2-15. Interrupt Vector Addresses

### 2.2.5 Status

The status character(s) is a series of bits that are input to the computer, upon request of the computer, from a selected Distributed I/O channel. The software uses the configuration of the status bits to determine the present operational conditions of the channel. The number of status bits and the meaning of the status bits varies with each Intelligent Cable/peripheral pair. Refer to Section 3 for a detailed description of the individual status bits for each Intelligent Cable. Most Intelligent Cable/peripheral pairs input a maximum of eight status bits which is the maximum number of status bits transferred into the computer with a single status request. When more than eight status bits are used, multiple status requests must be issued to input all the status bits.

Status is requested by the computer using the input A or X instructions described previously. The I/O Distributor address and the channel address select the specific Intelligent Cable/peripheral pair and the function code (LSB = 1) specifies the input status request. Up to eight status bits are input to the corresponding bit locations, 0 through 7 of A or X register. When multiple status request are required, each group of eight status bits is input as a separate status byte.

Status is normally requested as part of the End-of-Block subroutine.

## 2.3 NON-INTERRUPT/NON-DMA OPERATION

Operation of the Distributed I/O System in a non-interrupt/non-DMA programming environment is possible when sufficient care is taken in preparing the software. The main problem in attempting non-interrupt/non-DMA I/O is bypassing or simulating the functions normally performed by the I/O distributor. Since the functions that must be bypassed or simulated are dependent on the specific intelligent cable in use, very few general rules can be listed.

A function that must be bypassed with most Intelligent Cables is the timing of the data transfer. Under Auto I/O or DMA, data transfers occur whenever the intelligent cable request service by the I/O distributor. Under non-interrupt/non-DMA I/O, the software must continuously request channel status waiting for the specific configuration that indicates a data transfer is required. This configuration varies with each different intelligent cable. Refer to Section 3 for further details on the Intelligent Cable status characters.

A function that must be simulated with most Intelligent Cables is the acknowledgement of a data service request made by the Intelligent Cable. Since the Intelligent Cable generates the data service request, disabling interrupts or DMA does not suppress the data service requests. Since a pending data service request halts further PICOPROCESSOR firmware sequencing, the software must artificially acknowledge any requests that occur. With most Intelligent Cables, this can be accomplished by issuing a branch command word to the Intelligent Cable. The PICOPROCESSOR firmware ignores the branch address field and uses the Branch command word as an acknowledgement of the data service request.

The Intelligent Cable falls into three categories as regards non-interrupt, non-DMA operation:

- Those cables where frequent use of non-interrupt/non-DMA operation was anticipated. These cables provide a means for suppressing service requests upon specific command.

- Those cables where occasional non-interrupt/non-DMA operation was anticipated. These cables require that software perform many of the I/O distributors functions.

- Those cables where operation speed or other considerations make non-interrupt/non-DMA operation impractical.

In all cases, careful study of the Intelligent Cable descriptions in Section 3 of this manual is required before attempting non-interrupt/non-DMA operation programming.

# SECTION 3

# STANDARD INTELLIGENT CABLES

## 3.1 INTRODUCTION

Standard Intelligent Cables are available with PICOPROCESSORS microprogrammed for operation with common peripheral devices and usually with the appropriate device mating connectors installed. Some of the devices interfaced with standard Intelligent Cables include a line printer, card reader, high speed paper tape punch, high speed paper tape reader, etc. In addition, General Purpose Intelligent Cables for both positive true and negative true interface logic are available. These have a PICO-PROCESSOR with two general-purpose operating modes and are supplied without a device connector. User's Microcoded versions of the General Purpose Intelligent Cables are available to meet unique interface requirements.

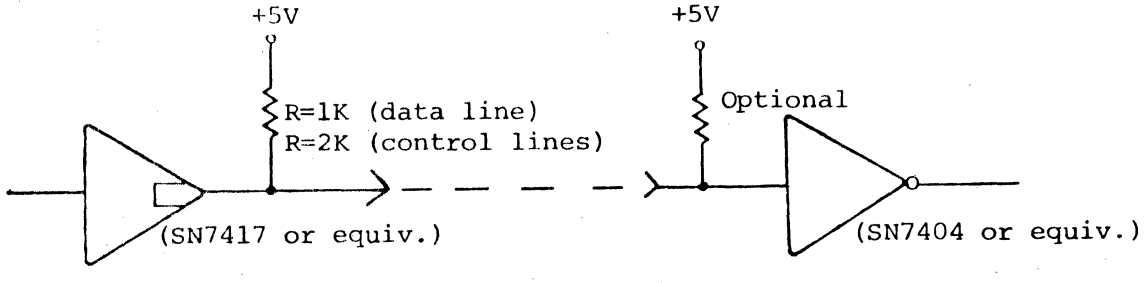
This section describes each standard Intelligent Cable including its firmware sequence, device interface lines and the mechanical details.

## 3.2 INTERFACE LOGIC

The interface logic used in several of the Intelligent Cables is common logic as described here. Interface logic that differs from the common logic is described under the specific Intelligent Cable description.

### 3.2.1. Output Logic

Intelligent Cable interface lines use standard TTL logic levels. Data lines are positive-true. Device control lines are either positive-true and/or negative-true depending on requirements of the particular device. Output drivers for the data and control lines are open-collector TTL buffers capable of sinking up to 32 ma at 5 volts.

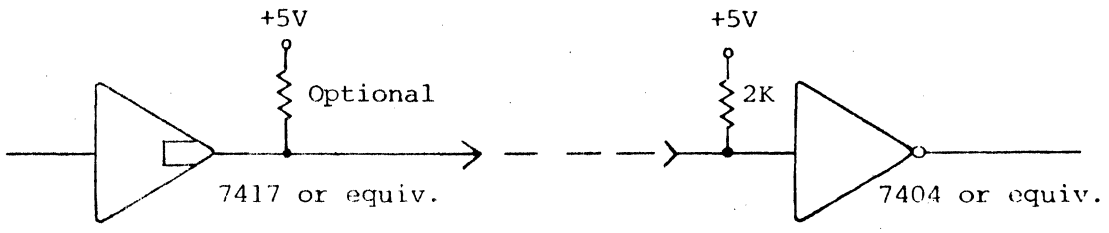



PicoProcessor Output

Typical Device Input

### 3.2.2 Input Logic

Input receivers use standard TTL logic levels, 0 to +5 volts, and present one TTL load with a 2K pull-up to +5 volts.




Typical Device Output PicoProcessor Input

## 3.3 STANDARD INTELLIGENT CABLE DESCRIPTIONS

### 3.3.1 Line Printer

#### 3.3.1.1 Description

Intelligent Cable 14631-01 controls the transfer of eight-bit positive-true parallel data to a Centronics Model 101, 306 or equivalent Line Printer. The PicoProcessor accepts a start command from the IOD, checks the status of the Line Printer then issues a data interrupt to start the data transfer. When all data is transferred or an error is encountered, the Pico-Processor generates an end-of-block interrupt and terminates the transfer operation.

#### 3.3.1.2 Physical Details

Cable Length
IOD to PicoProcessor, 10 1/2 feet
PicoProcessor to Line Printer, 1-1/2 feet
Standard Channel Number, 7 (Device Address Field = :FE)
Data Service Interrupt Vector Address, :F8
EOB Interrupt Address, :FC

#### 3.3.1.3 Line Printer Status Word

To control the transfer of data to the Line Printer, the PicoProcessor sequences through a series of operations based, in part, on the state of individual bits of the Line Printer Status Word. The Status Word, shown in figure 3-1, indicates certain operational and error conditions in the Line Printer. When the PicoProcessor receives an input instruction requesting device status (function control bit set to "1"), it immediately transfers the Status Word to the CPU on bits 0 through 5 of the data bus (all other bits are zero). Input status can be requested at any time, but it is usually done after an End-of-Block interrupt to determine the reason for termination. Individual bits of the Status Word are described in 3.3.1.5.2.

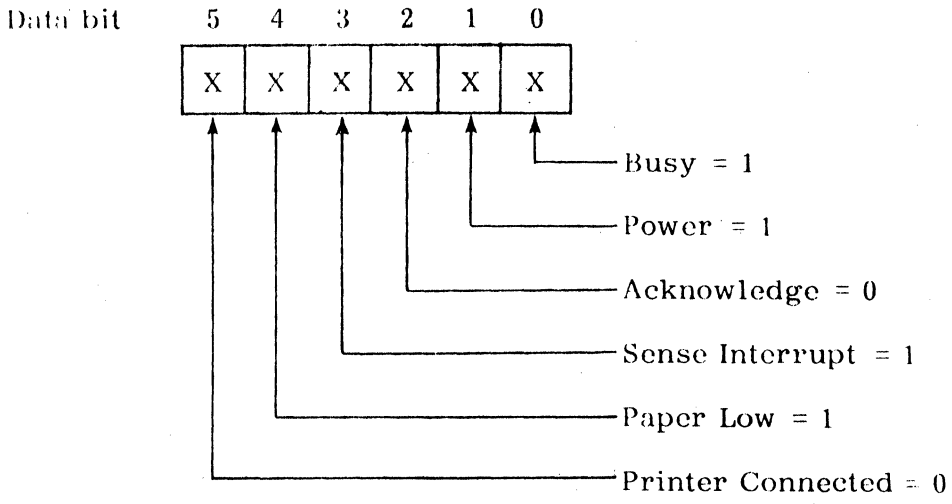



Figure 3-1. Line Printer Status Word

3.3.1.4 Operating Sequence

The PicoProcessor has 16 unique sequence addresses (:0 through :F).

When the PicoProcessor receives a Command Word with the Begin bit set to "1", it immediately begins operation at the sequence address specified by the Branch Address field of the Command Word. Standard software enters the sequence at one of the following sequence addresses:

| Sequence Address | Operation |
|---|---|
| :1 | Check Paper Low status; Start a Print Data operation |
| :2 | Start a Print Data operation (Skip Paper Low status check) |

The PicoProcessor terminates the operating sequence on any of the following conditions:

1. Paper Low
2. Data transfer completed
3. Line Printer power off
4. Line Printer cable disconnected

Details of the PicoProcessor operating sequence are shown in the flow chart (figure 3-2). The interface lines are described in 3.3.1.5. Following is a sequence description with each segment of the operation identified by name and sequence address (:0-:F). The yes/no decisions refer to the true/false state of a particular line as defined in 3.3.1.5, regardless of its logic level.

IDLE (:0)

Initially, the PicoProcessor is in the Idle state as a result of a Reset command or because of the completion of an end-of-block interrupt. A Begin command specifying a starting sequence address takes the PicoProcessor out of Idle State and the specified operation begins.

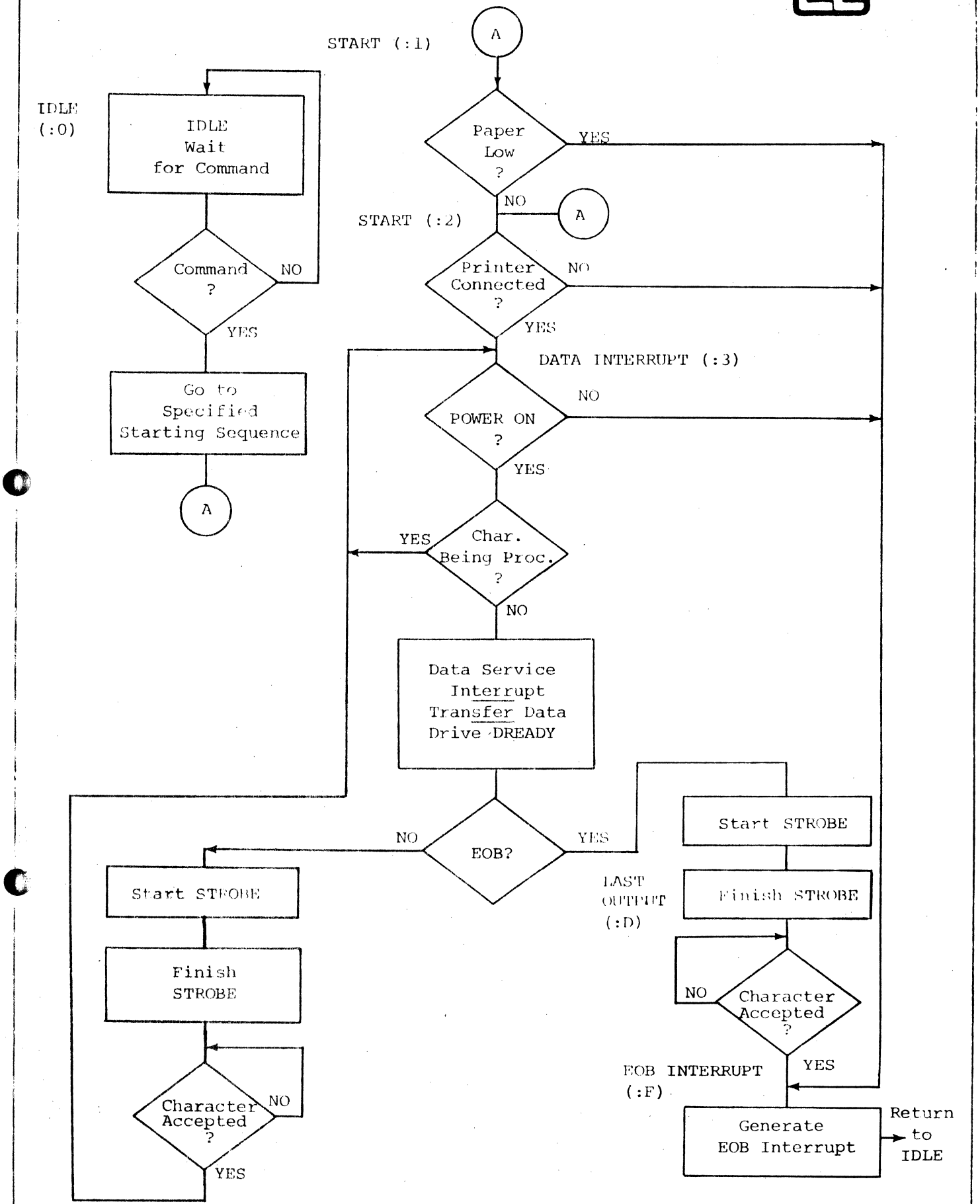



Figure 3-2. Firmware Sequence - Line Printer Intelligent Cable

START (:1)

The PicoProcessor begins the START sequence by first checking device status. Paper Low is checked first. If this line is true, the PicoProcessor generates an end-of-block interrupt to terminate the operation.

START (:2)

It is possible to start the sequence here if it is desired to skip the Paper Low status check. Proper device cabling is checked first. This status line indicates if the device is properly connected. If not, an end-of-block interrupt is generated. Otherwise, the PicoProcessor advances to the Data Interrupt sequence.

DATA INTERRUPT (:3)

The Power status line is checked to verify that power is applied to the device. If this status is false, an end-of-block interrupt is generated. Otherwise, the Acknowledge status line is monitored for a false state to indicate that the Printer is not processing a character. If this line is true, the PicoProcessor rechecks Power and then monitors the Acknowledge line again. It repeats this until the Acknowledge line is false.

The PicoProcessor then generates a Data Service interrupt causing the automatic output instruction at the data interrupt vector location to be executed. The CPU places data on the data bus and increments the byte count and memory buffer address.

When the PicoProcessor generates a Data Service Interrupt, it also drives the DREADY (data ready) control line which is directly tied to bit 3 of the device status word (Sense Interrupt). This status line is used during Programmed I/O transfers to inform the CPU that the device is ready to transfer data.

The PicoProcessor activates the 500 ns Strobe line to enter the data into the Line Printer. When the Printer acknowledges the data transfer (Acknowledge line true), the PicoProcessor repeats the Data Interrupt operation starting with the Power check.

The Data Interrupt segment of the sequence is repeated until all data is transferred. The PicoProcessor then enters the Last Output operation.

LAST OUTPUT (:D)

The PicoProcessor generates a 500-nanosecond Strobe signal to transfer the last byte of data to the Printer, then waits for the Printer to acknowledge the data. When the data is acknowledged, the PicoProcessor generates an EOB Interrupt.

EOB INTERRUPT (:F)

The PicoProcessor generates an end-of-block interrupt to terminate operations when all data has been transferred or when a status error is detected at any point in the sequence. When the interrupt operation is completed, the IOD commands the PicoProcessor to return to the IDLE state and wait for the next Begin command.

3.3.1.5 Interface Description

Interface lines include one control line to the Line Printer and five status lines from the Line Printer to the PicoProcessor. In addition, one control line (DREADY) is tied directly to the Sense Interrupt status line (figure 3-3).

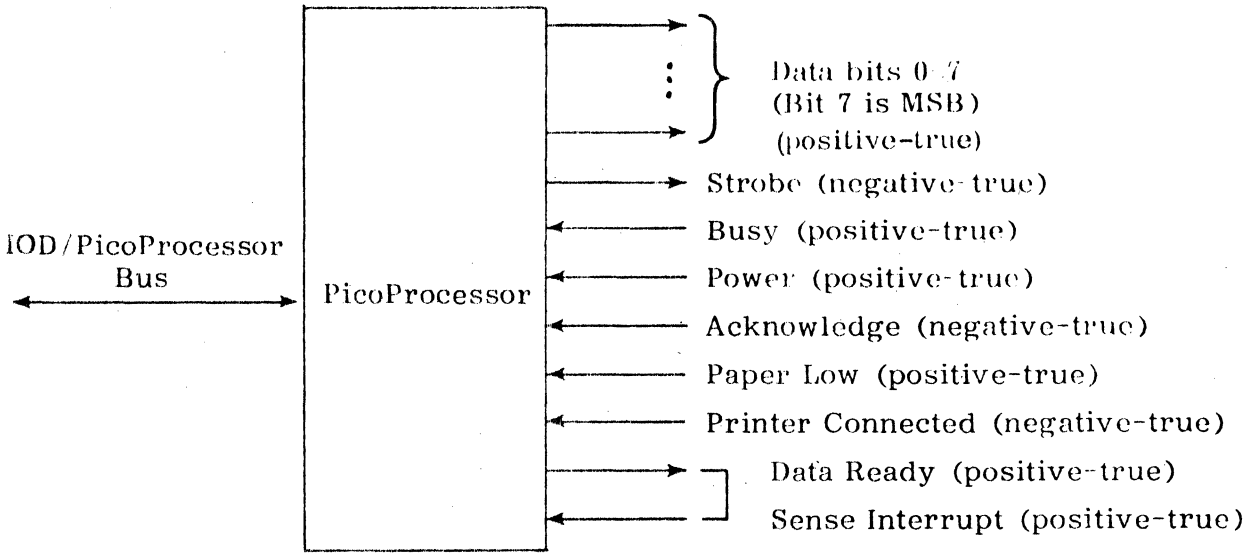



Figure 3-3. Line Printer Interface

3.3.1.5.1 Control Lines (to Line Printer).

1. Strobe. A 500-nanosecond negative-true signal generated by the PicoProcessor to enter data into the Line Printer.

2. Data Ready (positive-true). This line is driven by the PicoProcessor when a Data Service interrupt is generated. It is connected to the Sense Interrupt status line. See Sense Interrupt status description (paragraph 4 of "Status Line").

3. Reset (negative-true). This line is driven by the CPU RESET switch or under software control. Under software control, it is a 250 ns pulse. It is not used by the Line Printer.

3.3.1.5.2 Status Lines (from Line Printer).

1. Busy (positive-true). The Busy line remains true while the Line Printer is processing data from the previous data transfer. Acknowledge (negative true) will not be issued by the Line Printer until a few microseconds after Busy goes false. (Busy is not checked by the PicoProcessor during normal sequencing but is available for CPU status check.) If Printer is de-selected (off-line), the Busy line goes true until the Printer is selected.

2. Power (positive true). This status line is true whenever power is applied to the Line Printer. The PicoProcessor checks the status of this line before every data transfer to the device.

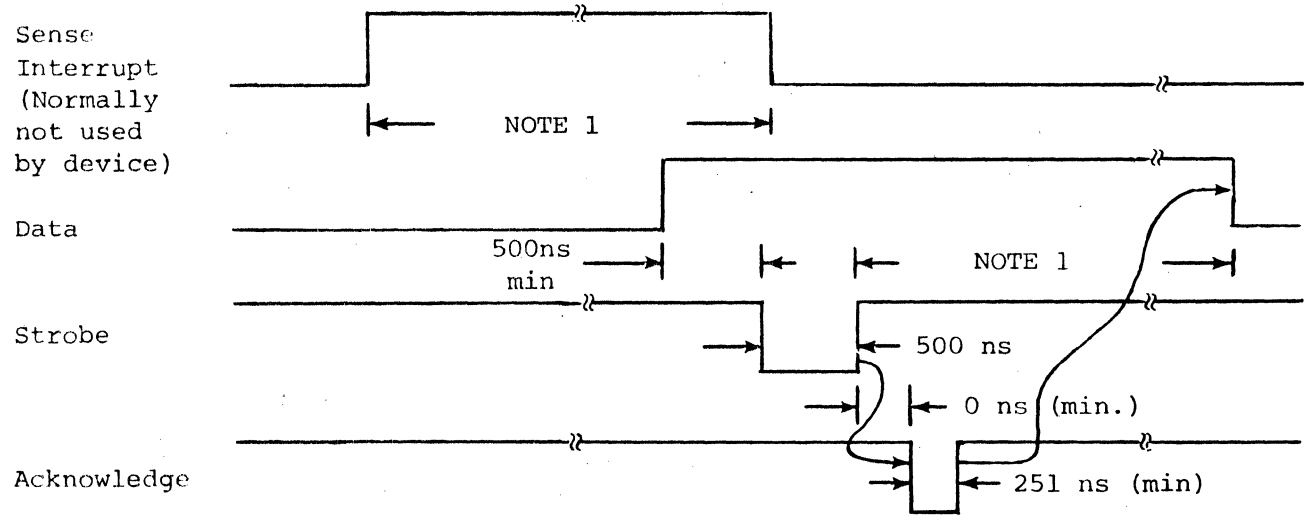
3. Acknowledge (negative-true). This line signifies that transferred data has been accepted by the Line Printer. Its relationship to the Strobe control line is shown in figure 3-4.

4. Sense Interrupt (positive-true). This line is true when the PicoProcessor generates a Data Service interrupt. It is used when operating in a non-interrupt I/O mode to inform the CPU that the device is ready to accept data. The device cable ties Data Ready and Sense Interrupt together at the device end of the cable.

5. Paper Low (positive-true). This line signifies that the paper supply in the Line Printer is low or nearly empty. The PicoProcessor checks the status of this line only once for every block of data transferred. The user can skip this status check if desired.

6. Printer Connected (negative-true). This line is true when Line Printer cabling is properly connected. This line is checked once for every block of data transferred.

Figure 3-4 shows the interface timing involved in transferring data to the printer. For further details, see the Centronics Line Printer instruction manual.




NOTE 1: Minimum time equal to the instruction time of the Auto I/O byte instruction plus 4 μsec. For instruction time, see the Appendix of the appropriate Computer Handbook.

Figure 3-4. Interface Timing - Line Printer PicoProcessor

3.3.1.5.3 Data Lines. The eight data lines to the Line Printer are positive true. Bit 0 is the least significant bit; bit 7 is the most-significant. Data is transferred in standard eight-bit ASCII characters.

3.3.1.6 Strapping Requirements - None Required

3.3.1.7 Device Cable Description

The Device Cable is 18 inches long and terminated on the PicoProcessor end with two 16-pin DIP plugs (P4 and P5; P6 is not used). The Line Printer end of the cable is terminated with a 36-pin connector that mates with J1 on the Line Printer.

Figure 3-6 lists all interface lines in the device cable and identifies the connectors used. The location of the mating connectors in the PicoProcessor is shown in figure 3-5.

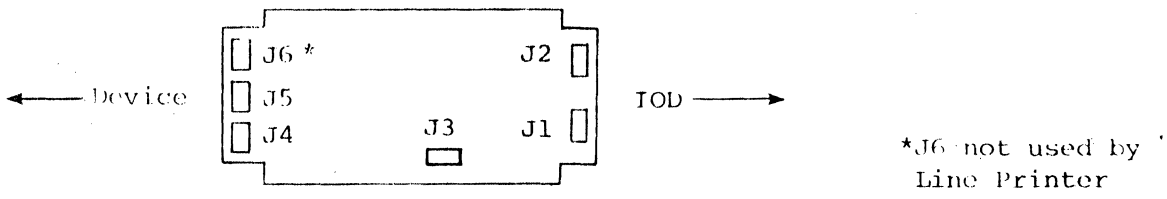



Figure 3-5. Connector Locations, Line Printer PicoProcessor

| PicoProcessor | | Description | Line Printer J1 pin | PicoProcessor | | Description | Line Printer J1 pin |
|---|---|---|---|---|---|---|---|
| Conn | Pin | | | Conn | Pin | | |
| P4 | 1 | Sense Interrupt * | 36 | P5 | 1 | Not used | -- |
| | 2 | Acknowledge | 10 | | 2 | Data bit 02 return | 22 |
| | 3 | Power | 18 | | 3 | Data bit 06 | 8 |
| | 4 | Busy | 11 | | 4 | Data bit 04 | 6 |
| | 5 | Reset (not used) | -- | | 5 | Data bit 02 | 4 |
| | 6 | Strobe | 1 | | 6 | Data bit 00 | 2 |
| | 7 | Control line (not used) | -- | | 7 | Printer connected | 28 |
| | 8 | Data Ready* | 36 | | 8 | Paper Low | 12 |
| | 9 | Data bit 06 return | 26 | | 9 | Data bit 00 return | 20 |
| | 10 | Data bit 07 return | 27 | | 10 | Data bit 01 return | 21 |
| | 11 | Strobe return | 19 | | 11 | Data bit 01 | 3 |
| | 12 | Ground (not used) | -- | | 12 | Data bit 03 | 5 |
| | 13 | Busy return | 29 | | 13 | Data bit 05 | 7 |
| | 14 | Data bit 04 return | 24 | | 14 | Data bit 07 (MSB) | 9 |
| | 15 | Data bit 05 return | 25 | | 15 | Data bit 03 return | 23 |
| P4 | 16 | Input Prime return | 30 | P5 | 16 | Not used | -- |

*P4-1 and P4-8 are tied together; J1-36 is not used by the Printer.

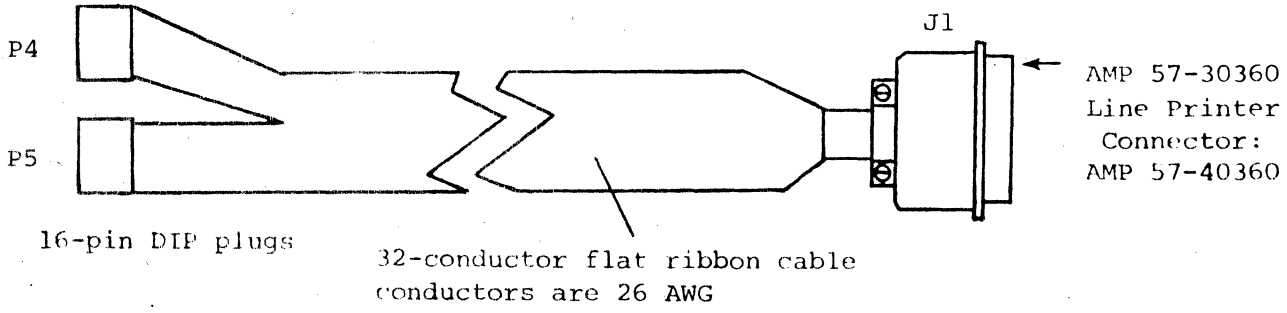



Figure 3-6. Cable Description - Line Printer

3.3.1.8 Programming Example

The Assembler Language statements shown in Table 3-1 demonstrate one method for using the Distributed I/O System. The technique is based on the programming in section 2.

The demonstration code falls into three distinct parts: Interrupt Locations, I/O Initiation and End-of-Block Service.

1. Interrupt Locations

The first part of the code defines and fills the standard locations associated with IOD channel to which the device's PICOPROCESSOR is connected. There are six locations involved of which two -- the Byte Count and the Buffer Location -- are dynamic for each I/O Initiation, while the remaining words can be fixed at program load time.

2. I/O Initiation

The second part of the code accomplishes the transfer of one physical record each time the user's main-line program executes this instruction:

CALL LPRINT

It is assumed that the calling program has previously set up these two words:

| | |
|---|---|
| BYTCNT | Number of bytes to be transferred |
| BUFADD | Word address of record buffer |

The demonstration code converts this information into the form required by an Automatic I/O instruction: negative byte count and byte address of the buffer minus one. Once these computations have been stored into the interrupt locations, the PicoProcessor is sent a command specifying:

Begin at Branch Address : 1

Notice that the Device Address used in this sequence is coded with the same Device Address used in the interrupt location, plus one. The assembled object code will indicate the transfer of a command, rather than of data, as shown in figure 2-3.

No attempt is made in the demonstration code to overlap record transfer with CPU activity. Instead, a dead loop is issued -- JMP $ -- until the PicoProcessor interrupts not to the Data Service interrupt location, but to the End-of-Block location. At this point control passes to the subroutine labelled EOB.

3. End-of-Block Service

The End-of-Block subroutine could perform error analysis, retry, etc. The demonstration code simply obtains device status from the PicoProcessor and passes status back to the main-line program through the A register.

Table 3-1. Programming Example - Line Printer

```
LPINT   EQU     :F8         STANDARD LINE-PRINTER INTERRUPT
LPDEVA  EQU     :FE         STANDARD DEVICE ADDRESS
        SPACE   1
        ABS     LPINT       INTERRUPT FOR DATA LOCATION
        AOB     LPDEVA      AUTO OUTPUT BYTE
        DATA    $-$         TO BE FILLED WITH BYTE COUNT
        DATA    $-$         TO BE FILLED WITH BUFFER ADDRESS-1
        DATA    Ø

*                           INTERRUPT FOR END-OF-BLOCK
        JST     *$+1        CALL END-OF-BLOCK ROUTINE
        DATA    EOB

LPRINT  ENT                 ENTRY POINT FOR LP DIRVER
        LDA     BYTCNT      BYTE COUNT FOR MESSAGE
        NAR                 AOB INSTRUCTIONNEEDS NEGATIVE
        STA     LPINT+1     PUT IN AOB INSTRUCTION
        LDA     BUFADD      ADDRESS (WORD) OF BUFFER
        LLA     1           AOB INSTRUCTION NEEDS BYTE ADDRESS
        SAI     1           STARTS AT -1
        STA     LPINT+2     PUT IN AOB INSTRUCTION
        LDA     =:0210      WORD TO START PICOPROCESSOR
        OTA     LPDEVA+1    SEND COMMAND TO PICO
        JMP     $           WAIT FOR END-OF-BLOCK

EOB     ENT                 END-OF-BLOCK INTERRUPT SUBROUTINE
        INA     LPDEVA+1    INPUT STATUS
        RTN     LPRINT      RETURN TO CALLER WITH
*                           STATUS IN A REGISTER
```

### 3.3.2 Card Reader

#### 3.3.2.1 Description

Intelligent Cable 14631-02 controls the input of 12-bit positive-true parallel data from a Documation M-200, M-600, M-1000 or M-1200 Card Reader. The cable's PicoProcessor accepts an input command from the IOD, checks the status of the Card Reader, generates a command to start the card through the Reader, then generates two Data Service interrupts per column to transfer data to the CPU, one byte at a time. When all data is transferred or an error is detected, the PicoProcessor generates an End-of-Block interrupt to terminate the operation.

#### 3.3.2.2 Physical Details

Cable Lengths:

IOD to PicoProcessor, 10-1/2 feet
PicoProcessor to Card Reader, 1-1/2 feet

Standard Channel Number, 0 (Device Address field = :F0)
Standard Data Service Interrupt Address, :C0
Standard End-of-Block Interrupt Address, :C4

#### 3.3.2.3 Card Reader Status Word

To control the transfer of data from the Card Reader, the PicoProcessor sequences through a series of operations based, in part, on the state of individual bits of the Card Reader Status Word. The Status Word, shown in figure 3-7, indicates certain operational and error conditions in the Card Reader. When the PicoProcessor receives an input instruction requesting device status (function control bit set to "1"), it immediately transfers the entire Status Word to the CPU on bits 0 through 5 of the data bus. Input status can be requested at any time, but it is usually done after an End-of-Block interrupt to determine the reason for termination. Individual status bits are described in 3.3.2.5.2.

Certain status bits are not checked by PicoProcessor firmware but are available for status checking by software. These include Error, Sense Interrupt and Hopper Check.

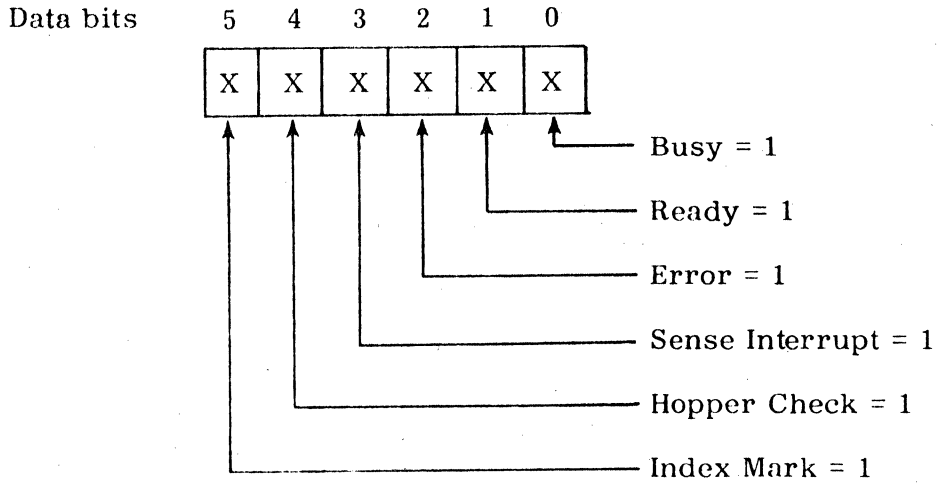



Figure 3-7. Card Reader Status Word

3.3.2.4 Operating Sequence

The PicoProcessor has 16 unique sequence addresses (:0 thru :F). When the PicoProcessor receives a Command Word with the Begin bit set to "1", it immediately begins operation at the sequence address specified by the Branch Address field of the Command Word. Standard software enters the sequence at the following address:

| Sequence Address | Operation |
|---|---|
| :1 | Read one card; generate EOB interrupt if an error exists or when all data transferred. |

Details of the PicoProcessor operating sequence are shown in the flow chart (figure 3-8). All interface lines are described in 3.3.2.5. Following is a sequence description with each segment of the operation identified by name and sequence address (:0 thru :F). The yes/no decisions refer to the true/false state of a particular line as defined in 3.3.2.5, regardless of its logic level.

IDLE (:0)

Initially, the PicoProcessor is in the Idle state as a result of a Reset command or because of the completion of an End-of-Block interrupt. A Begin Command specifying a start address takes the PicoProcessor out of the Idle State.

START (:1)

The PicoProcessor waits for the Busy line to go false indicating that the Card Reader is not processing a card. It then generates a Pick command to start the next card through the Reader and checks the Ready status line. If Ready is true, it indicates that Card Reader is ready to read data. If false, the PicoProcessor generates an immediate End-of-Block interrupt to terminate the operation.

The PicoProcessor again checks the Busy line. It waits for the line to go true indicating that the Reader has started looking for data.

NEXT COLUMN (:7)

The PicoProcessor continues to monitor the Busy line. If the line is false, the PicoProcessor generates an immediate End-of-Block interrupt. If the Busy line remains true, it indicates that the Reader has begun to read data from the card.

The PicoProcessor detects the presense of a column index mark and generates two Data Service interrupts to the CPU. It detects the Index Mark by first checking for a true level, then a false level on the Index Mark status line.

When the Index Mark is detected, the PicoProcessor generates a Data Service interrupt causing the Automatic Input instruction at the interrupt location to be executed. One byte of data (MSB) is transferred to the CPU and the byte count and memory buffer are incremented.

When the PicoProcessor generates a Data Service interrupt, it also drives the DREADY (data ready) control line. The signal is tied to the Sense Interrupt status line (bit 3 of the device status word). This status line informs the CPU, when operating in non-interrupt I/O mode, that data can be input from the Card Reader.

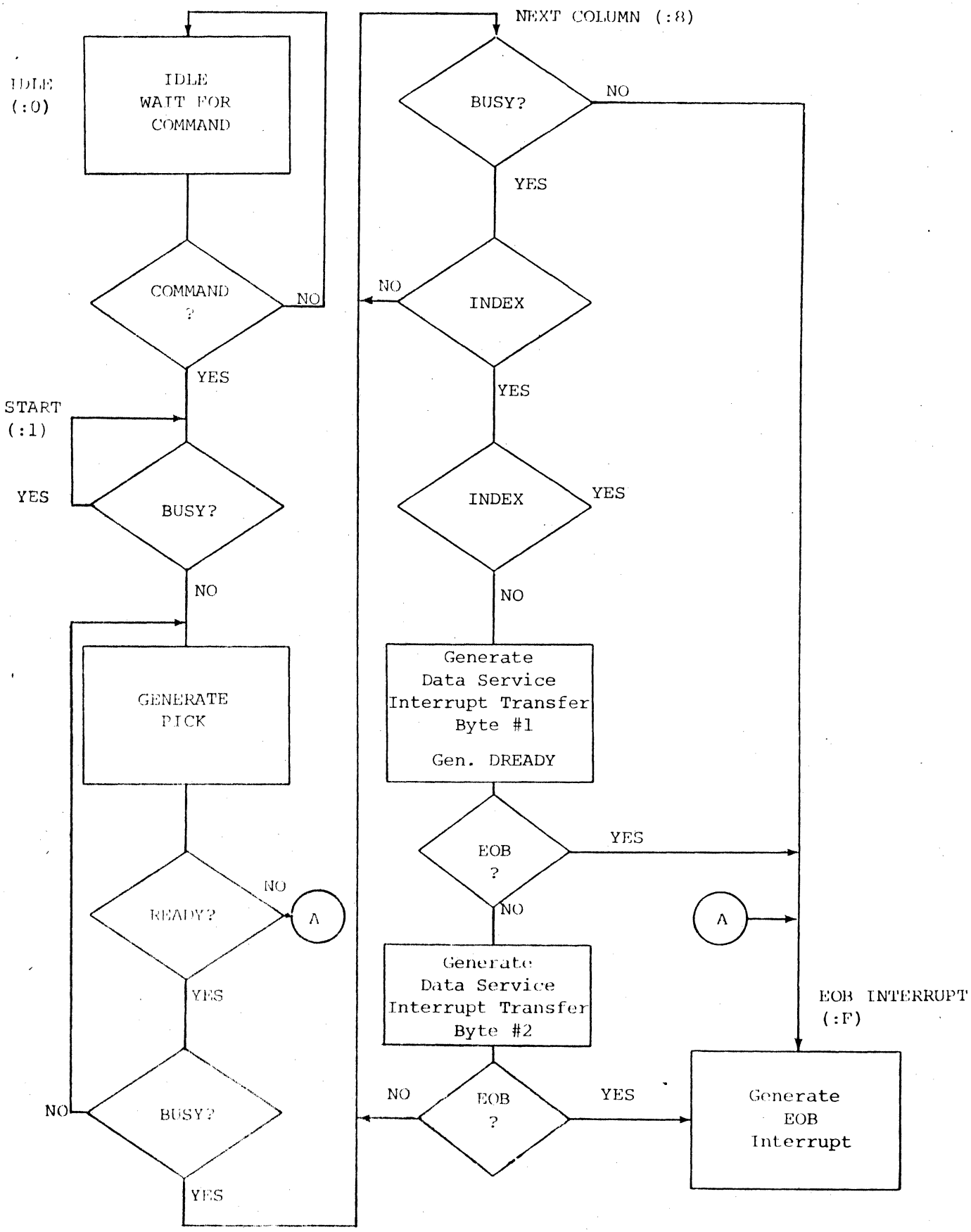



Figure 3-8. Firmware Sequence - Card Reader Intelligent Cable

If the data transfer did not increment the byte count to zero, a second Data Service interrupt is generated to transfer the second byte (LSB) of data to the CPU. Since the Card Reader is a 12-bit device, the upper four bits of the first byte are filled with zeros.

When the second byte is transferred, the PicoProcessor again checks for byte count=0. If byte count does equal zero, the PicoProcessor generates an End-of-Block interrupt. If the byte count does not equal zero, the PicoProcessor returns to the Next Column sequence where it checks Busy status, waits for the Index Mark and then generates more Data Service interrupts. This continues until all data has been transferred.

EOB INTERRUPT (:F)

The PicoProcessor generates an EOB interrupt to terminate the data transfer operation when all data has been transferred or when a status error is detected at any point in the sequence. When the EOB interrupt is serviced, the IOD commands the PicoProcessor to return to the Idle state and await the next Begin command.

3.3.2.5 Interface Description

Interface lines between the PicoProcessor and the Card Reader include 16 data lines, one control line to the Card Reader and 5 status lines from the Card Reader (figure 3-9). In addition, one control line (DREADY) is tied directly to the Sense Interrupt status line. All lines are positive-true. Since the Card Reader is a 12-bit device, the four most-significant data bits are tied to ground.

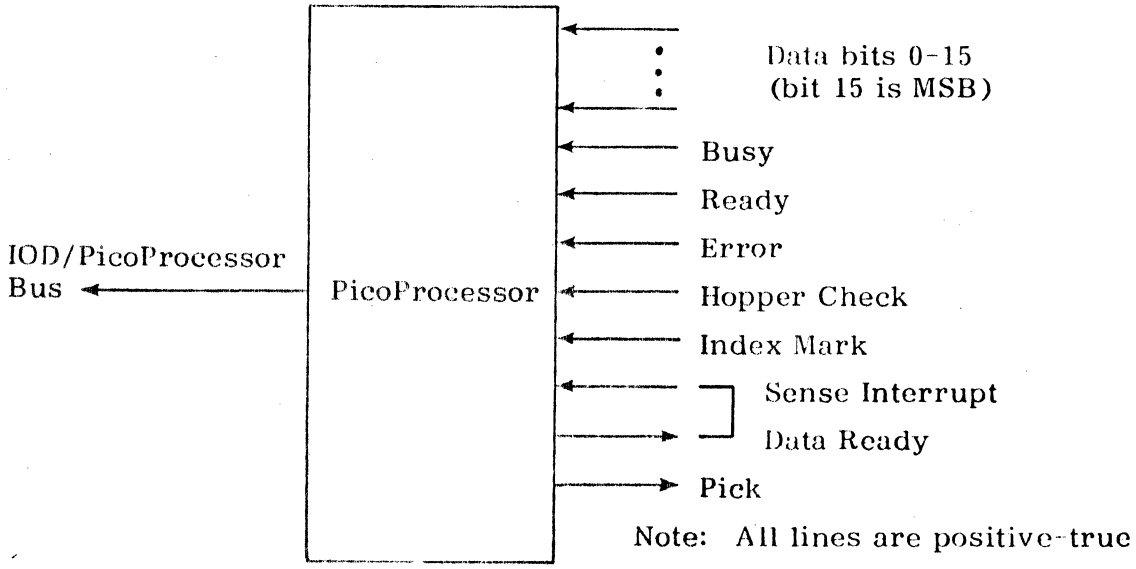



Figure 3-9. Card Reader Interface

3.3.2.5.1 Control Lines (To Card Reader).

1. Data Ready. This line is driven by the PicoProcessor when a Data Service interrupt is generated. It is connected to the Sense Interrupt status line. See Sense Interrupt status line description (paragraph 4 of "Status Lines").

2. Pick. The PICK command from the PicoProcessor starts a card moving through the Card Reader.

3. Reset. This negative-true line is driven by the CPU RESET switch or under software control. Under software control, it is a 250 ns pulse. This line is not used by the Card Reader.

3.3.2.5.2 Status Lines (From Card Reader).

1. Busy. This line is true when the Card Reader is reading or processing data and cannot accept a PICK command.

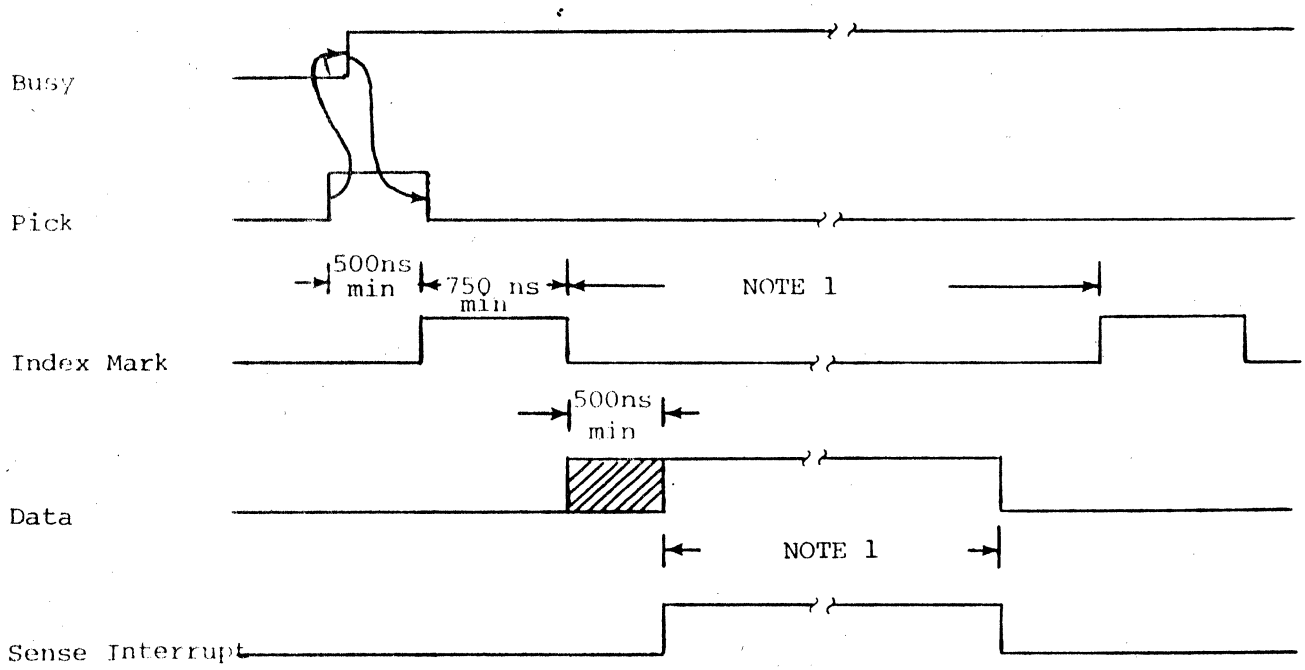
2. Ready. This line is true to indicate that the Card Reader is clear of errors and ready to accept a PICK command. This line is checked only once for every block of data transferred.

3. Error. This line is true to indicate that an error condition exists in the Card Reader. It is not checked by PicoProcessor firmware, but is available for software interrogation.

4. Sense Interrupt. Sense Interrupt is true (driven by DREADY) when the PicoProcessor generates a Data Service interrupt. It is used when operating in a non-interrupt mode to inform the CPU that data in the Card Reader is ready to be transferred.

5. Hopper Check. This line is true when the card hopper is empty of cards. It is not checked by PicoProcessor firmware, but is available for software interrogation.

6. Index Mark. An Index Mark is generated by the Card Reader once for each column of data as the card moves through the Reader.




NOTE 1: Minimum time equal to twice the instruction time of the Auto I/O byte instruction plus 8 μsec. For instruction time, see the Appendix of the appropriate Computer handbook.

Figure 3-10. Interface Timing - Card Reader Intelligent Cable

3.3.2.5.3 Data Lines. The 12 data lines to the Card Reader are positive-true. The data is transferred in two bytes, most-significant byte first. The 12 data columns of the card are right-justified into the memory word with the upper (unused) four bits zero:

| Data Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Card Data Column | zero | | | | 12 | 11 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Figure 3-10 shows the interface timing. For additional details, see the Documation Card Reader instruction manual.

3.3.2.6 Strapping Requirements

The Sequence Select factory-installed jumper is installed in connector J3 across pins 5 and 12. The location of J3 is shown in figure 3-11.

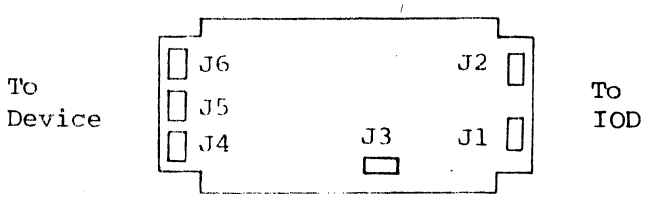



Figure 3-11. Connector Locations, Card Reader PicoProcessor

3.3.2.7 Device Cable Description

The device cable is 18-inches long and terminated on the PicoProcessor end with three 16-pin DIP plugs (P4, P5 and P6). The Card Reader end of the cable is terminated with a 38-pin connector that mates with Card Reader connector J1.

Figure 3-12 lists all interface lines in the device cable and identifies the connectors used. The location of the mating connectors on the PicoProcessor is shown in figure 3-11.

| PicoProcessor Conn | PicoProcessor Pin | Description | Card Reader J1 pin | PicoProcessor Conn | PicoProcessor Pin | Description | Card Reader J1 pin |
|---|---|---|---|---|---|---|---|
| P4 | 1 | Sense Interrupt* | -- | P5 | 9 | Hopper Check return | PP |
| | 2 | Error | HH | | 10 | Motion Check return | RR |
| | 3 | Ready | BB | | 11 | } Not used | |
| | 4 | Busy | MM | | thru | | |
| | 5 | Reset (not used) | -- | | 14 | | |
| | 6 | Pick | LL | | 15 | Row 11 Data return | F |
| | 7 | Control bit 1 (not used) | -- | P5 | 16 | Row 8 Data | Y |
| | 8 | Data Ready* | -- | P6 | 1 | Row 6 Data return | W |
| | 9 | Row 0 Data return | H | | 2 | Row 4 Data return | S |
| | 10 | Row 1 Data return | J | | 3 | Row 2 Data return | P |
| | 11 | Pick return | SS | | 4 | Row 11 Data | B |
| | 12 | Row 8 Data return | CC | | 5 | Row 1 Data | D |
| | 13 | Busy return | TT | | 6 | Row 3 Data | L |
| | 14 | Ready return | FF | | 7 | Row 5 Data | N |
| | 15 | Error return | NN | | 8 | Row 7 Data | V |
| P4 | 16 | Index Mark return | EE | | 9 | Row 6 Data | U |
| P5 | 1 | Row 9 Data | Z | | 10 | Row 4 Data | M |
| | 2 | Row 12 Data return | E | | 11 | Row 2 Data | K |
| | 3 | } Not used | | | 12 | Row 0 Data | C |
| | thru | | -- | | 13 | Row 12 Data | A |
| | 6 | | | | 14 | Row 3 Data return | R |
| | 7 | Index Mark | AA | | 15 | Row 5 Data return | T |
| P5 | 8 | Hopper Check | JJ | P6 | 16 | Row 7 Data return | X |

*Pins 1 and 8 of P4 are tied together.

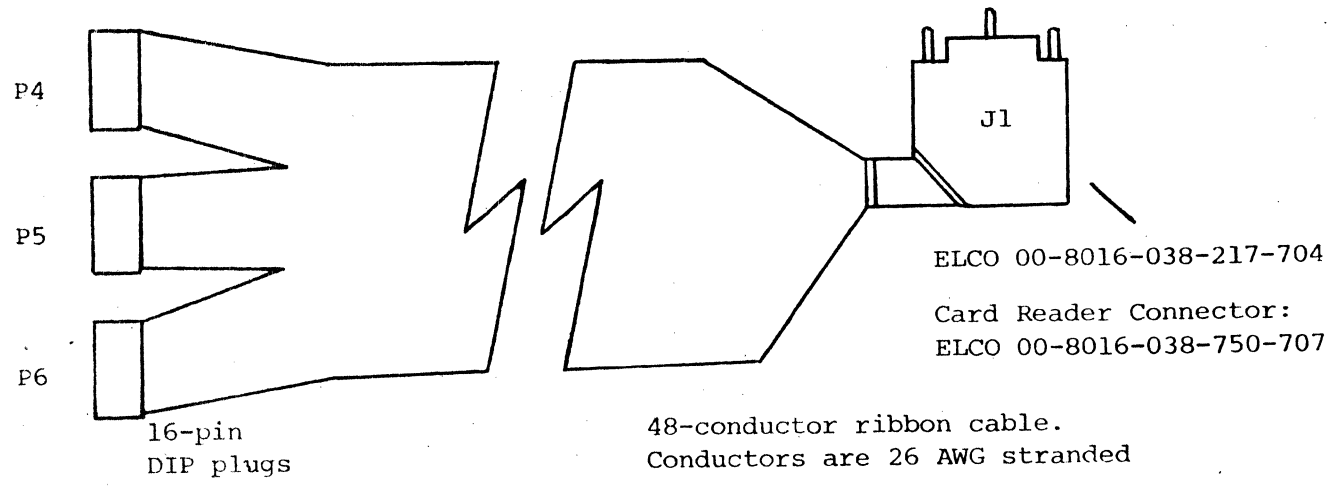



Figure 3-12. Device Cable - Card Reader

3.3.2.8 Programming Example

The Assembler Language statements shown in Table 3-2 demonstrate one method for using the Distributed I/O System. The technique is based on the programming information in Section 2.

The demonstration code falls into three distinct parts: Interrupt Locations, I/O Initiation and End-of-Block Service.

1. Interrupt Locations

The first part of the code defines and fills the standard locations associated with IOD channel to which the device's PicoProcessor is connected. As shown in figure 2-4, there are six locations involved of which two -- the Byte Count and the Buffer Location -- are dynamic for each I/O Initiation, while the remaining words can be fixed at program load time.

2. I/O Initiation

The second part of the code accomplishes the transfer of one physical record each time the user's main-line program executes this instruction:

```
CALL CARD
```

It is assumed that the calling program has previously set up these two words:

```
COLCNT    Number of bytes to be transferred
BUFADD    Word address of record buffer
```

The demonstration code converts this information into the form required by an Automatic I/O instruction: negative byte count and byte address of the buffer minus one. Once these computations have been stored into the interrupt locations, the PicoProcessor is sent a command specifying:

```
Begin at Branch Address : 1
```

Notice that the Device Address used in this sequence is coded with the same Device Address used in the interrupt location, plus one. The assembled object code will indicate the transfer of a command, rather than of data.

No attempt is made in the demonstration code to overlap record transfer with CPU activity. Instead, a dead loop is issued -- JMP $ -- until the PicoProcessor interrupts not to the Data Service interrupt location, but to the End-of-Block location. At this point control passes to the subroutine labelled EOB.

3. End-of-Block Service

The End-of-Block subroutine could perform error analysis, retry, etc. The demonstration code simply obtains device status from the PicoProcessor and passes status back to the main-line program through the A register.

Table 3-2. Programming Example - Card Reader

```
CRINT   EQU     :C#         STANDARD CARD READER INTERRUPT
CRDEVA  EQU     :F#         STANDARD CARD READER DEVICE ADDRESS
*
        ABS     CRINT       INTERRUPT FOR DATA LOCATION
        AIB     CRDEVA      AUTO INPUT BYTE
        DATA    *-$         TO BE FILLED WITH COUNT

        DATA    *-$         TO BE FILLED WITH BUFFER ADDRESS-1
        DATA    0
*               INTERRUPT FOR END-OF-BLOCK
        JST     *$+1        CALL END OF BLOCK ROUTINE
        DATA    EOB
*
CARD    ENT                 ENTRY POINT FOR CARD-READER DRIVER
        LDA     COLCNT      # OF COLUMNS WANTED
        LLA     1           TWO BYTES/COLUMN
        NAR                 AIB INSTRUCTION NEEDS NEGATIVE
        STA     CRINT+1     PUT IN AIB INSTRUCTION
        LDA     BUFADD      ADDRESS (WORD) OF BUFFER
        LLA     1           AIB INSTRUCTION NEEDS BYTE ADDRESS
        SAI     1           STARTS AT -1
        STA     CRINT+2     PUT IN AIB INSTRUCTION
        LDA     =:#210      WORD TO START PICOPROCESSOR
        OTA     CRDEVA+1    SEND TO PICO
        JMP     $
*
EOB     ENT     END-OF-BLOCK INTERRUPT SUBROUTINE
        INA     CRDEVA+1    INPUT STATUS
        RTN     CARD        RETURN TO CALLER WITH
*                           STATUS IN A REGISTER
```

### 3.3.3 High-Speed Paper Tape Reader

#### 3.3.3.1 Description

Intelligent Cable 14631-03 controls the transfer of eight-bit positive-true parallel data from a Remex RAB6375BA1/661/551/U000 Reader (see 3.3.4.1) to the computer at the rate of 300 characters per second. The PicoProcessor accepts a start command from the IOD, checks device status, then issues a Data Service interrupt to start the data transfer. When the transfer is complete or an error detected, the PicoProcessor generates an end-of-block interrupt to terminate the transfer.

#### 3.3.3.2 Physical Details

Cable Lengths:
IOD to PicoProcessor, 4 feet
PicoProcessor to Reader, 1-1/2 feet.

Standard Channel Number, 5 (Device Address field=:FA)
Standard Data Service Interrupt Address, :E8
Standard EOB Interrupt Address, :EC

#### 3.3.3.3 Paper Tape Reader Status Word

To control the transfer of data from the Paper Tape Reader, the PicoProcessor sequences through a series of operations based in part, on the state of individual bits of the Paper Tape Reader Status Word. The Status Word, shown in figure 3-13, indicates certain operational conditions in the Paper Tape Reader. When the PicoProcessor receives an input instruction requesting device status (function control bit set to "1"), it immediately transfers the entire Status Word to the CPU on bits 0 through 5 of the data bus. Input status can be requested at any time, but it is usually done after an End-of-Block interrupt to determine the reason for termination. Individual bits of the Status Word are described in 3.3.3.5.2.

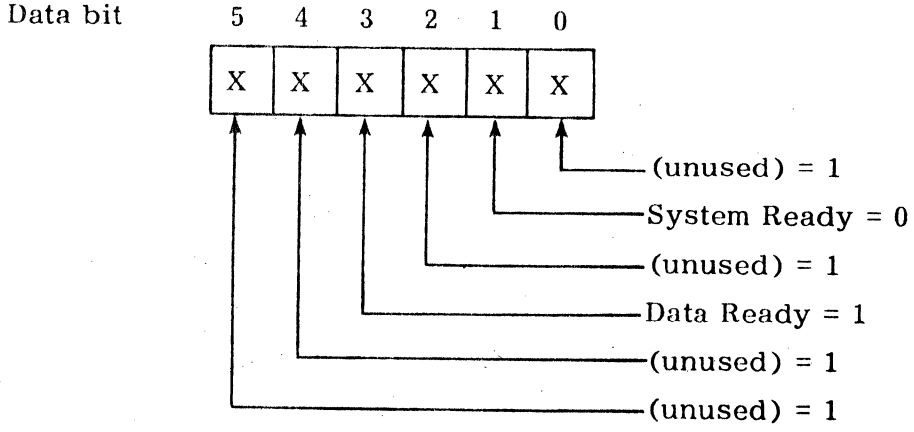



Figure 3-13. Paper Tape Reader Status Word

#### 3.3.3.4 Operating Sequence

The PicoProcessor has 16 unique sequence addresses (:0 thru :F). When the PicoProcessor receives a Command Word with the Begin bit set to "1", it immediately begins operation at the sequence address specified by the Branch Address field of the Command Word. Standard software enters the sequence at one of the following sequence addresses.

| Sequence Address | Operation |
|---|---|
| :1 | Read Tape, Left to Right |
| :A | Read Tape, Right to Left |

Details of the PicoProcessor operating sequence are shown in the flow chart (figure 3 14). Interface lines are described in 3.3.3.5. Following is a sequence description with each segment of the operation identified by name and sequence address (:0 thru :F). The yes/no decisions refer to the true/false state of a particular line as defined in 3.3.3.5, regardless of the logic level.

IDLE (:0)

The PicoProcessor is in the Idle state as a result of a Reset command or because of the completion of an end-of-block interrupt. A Begin command selects a Start sequence for either a Drive Right or Drive Left operation. Since the operation sequence is the same in both tape directions, they are combined in the flow chart to Start Right/Left.

START RIGHT/LEFT (:1/:A)

A Drive Right or Drive Left command is issued to start tape motion and Data Ready status is monitored. When the Data Ready line goes false, the reader head is moving and "between characters" on the tape. System Ready status is then checked. This line is true if power is applied to the Reader system and the Reader RUN/LOAD switch is in the RUN position. If false, the PicoProcessor immediately generates an end-of-block interrupt to terminate the operation. Otherwise, the PicoProcessor monitors the Data Ready status line again, this time looking for a true level on the line to indicate that the Reader head is "on character". When a positive response is obtained, the PicoProcessor generates a Data Service interrupt causing the Automatic input instruction at the data interrupt location to be executed. Data from the Reader is transferred into the CPU and the byte count and memory buffer are incremented. A line from the CPU is activated if the byte count increments to zero to signal the PicoProcessor (via the IOD) that all data has been transferred (end-of-block).

If the byte count does not increment to zero, the PicoProcessor again generates Right/Left Drive, checks the status of Data Ready, checks System Ready and generates another Data Service Interrupt. This operation is repeated until all data has been transferred. The PicoProcessor then enters the EOB Sequence.

EOB INTERRUPT (:F)

The PicoProcessor generates an EOB interrupt to terminate operations when all data has been transferred or when a status error is detected at any point in the sequence. When the EOB interrupt is serviced, the CPU commands the PicoProcessor (via the IOD) to return to the Idle state and wait for the next command.

### 3.3.3.5 Interface Description

Interface lines between the PicoProcessor and the HSPT Reader consist of eight data lines, two control lines to the Reader and two status lines from the Reader to the PicoProcessor (figure 3-15).

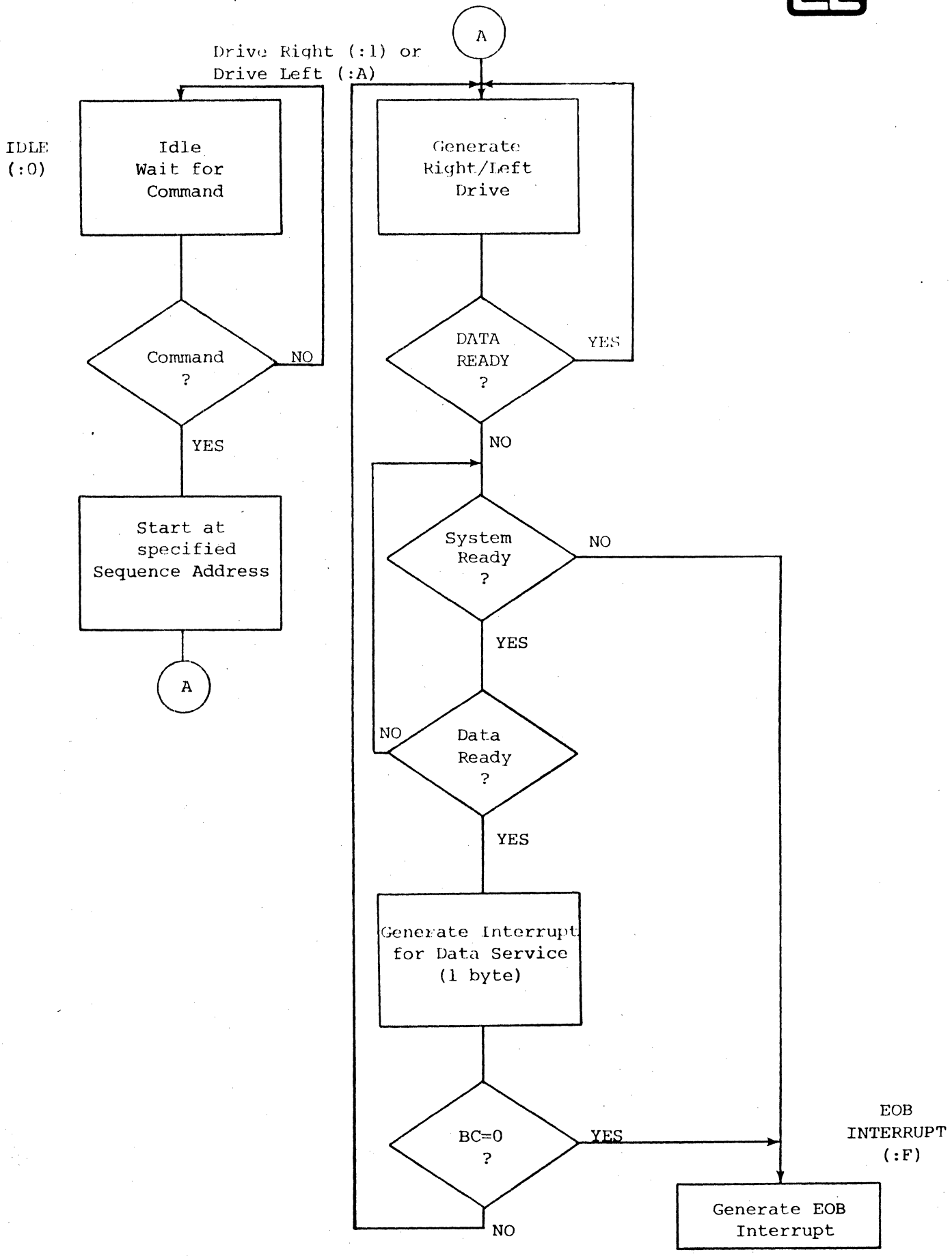



Figure 3-14. Firmware Sequence - HSPT Reader Intelligent Cable

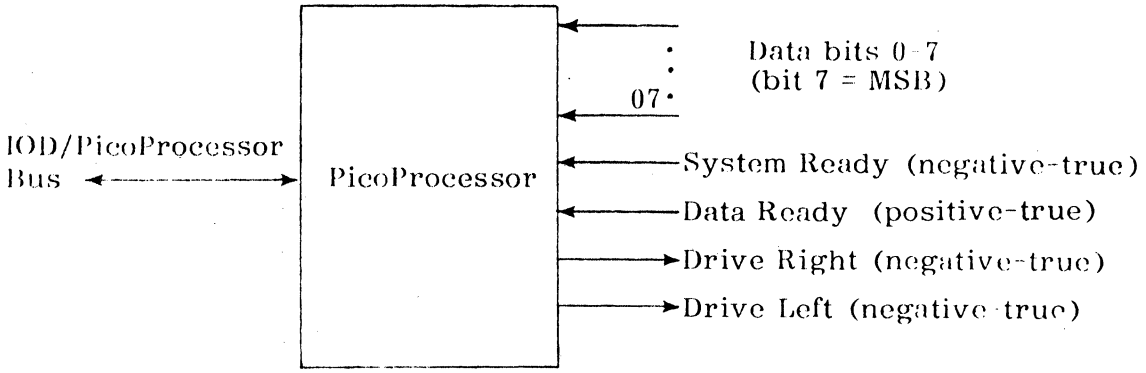



Figure 3-15. Paper Tape Reader Interface

3.3.3.5.1 Control Lines (to Device).

1. Drive Right. This line (negative-true) is driven by the Intelligent Cable to start the tape moving in the reader in the right-hand direction.

2. Drive Left. This line (negative-true) is driven by the Intelligent Cable to start the tape moving in the reader in the left-hand direction.

3. Reset. This negative-true line is driven by the CPU RESET switch or under software control. Under software control, it is a 250 ns pulse. It is not used by the Paper Tape Reader.

3.3.3.5.2 Status Lines (from Device).

1. Data Ready (positive-true). This line, when true, indicates that the data track outputs are in the "on character" position. When false, this line indicates the "between character" position where data outputs have no significance.

2. System Ready (negative-true). This line, when true, indicates that power is applied to the Reader and it is in the Run mode. It is false when the Reader is in the Load mode or if the Reader is out of tape. This line is checked before each data transfer.

Other status bits are not used by the Paper Tape Reader.

3.3.3.5.3 Data Input Lines. The eight data input lines (00 through 07) are positive-true. Bit 07 is the most-significant bit. The data is input from the Reader in standard eight-bit ASCII characters.

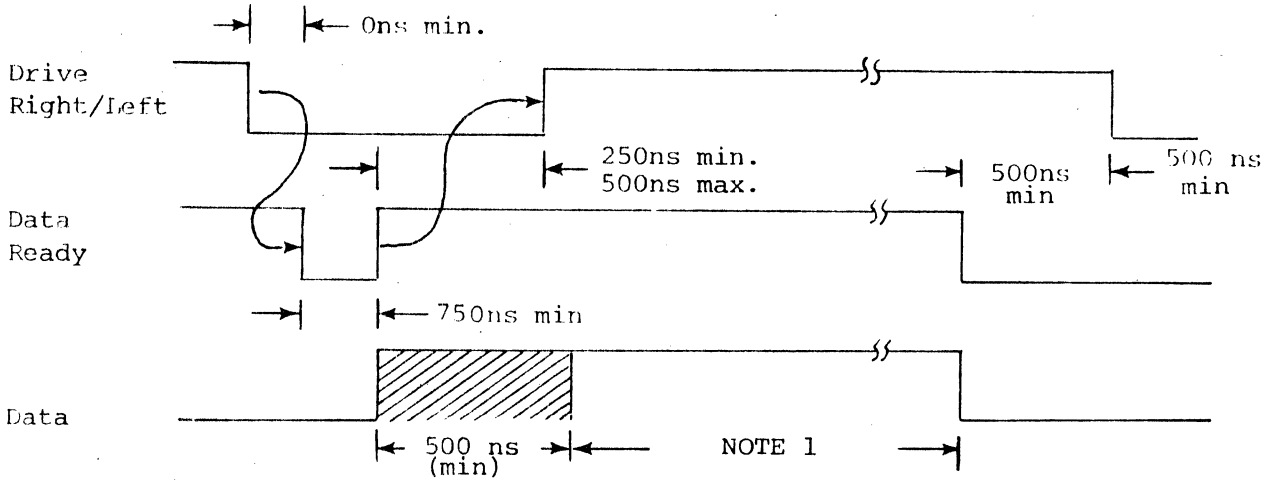
Figure 3-16 shows the interface timing. For additional details, see the Remex HSPT Reader/Punch instruction manual.

3.3.3.6 Strapping Requirements - None Required.

3.3.3.7 Device Cable Description

The Device cable is 18-inches long and terminated on the PicoProcessor end with three 16-pin DIP plugs (P4, P5, and P6). The Paper Tape Reader end of the cable is terminated with a 25-pin connector that mates with connector J2 on the Paper Tape Reader.

Figure 3-18 lists all interface lines in the device cable and identifies the connectors used. The location of mating connectors on the PicoProcessor is shown in figure 3-17.




NOTE 1: Minimum time equal to the instruction time of the Auto I/O byte instruction plus 4 μsec. For instruction time, see the Appendix of the appropriate Computer Handbook.

Figure 3-16. Interface Timing-HSPT Reader Intelligent Cable

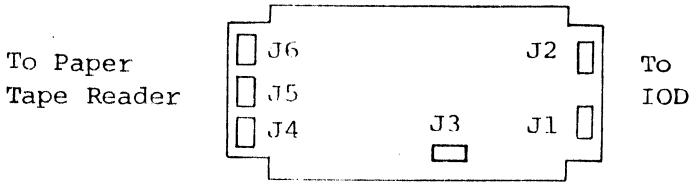



Figure 3-17. Connector Locations, HSPT Reader PicoProcessor

| PicoProcessor Conn | PicoProcessor Pin | Description | Reader/ Punch J2 pin | PicoProcessor Conn | PicoProcessor Pin | Description | Reader/ Punch J2 pin |
|---|---|---|---|---|---|---|---|
| P4 | 1 | Data Ready | 9 | P5 | 7 | Status bit 5 (not used) | - |
| ↑ | 2 | Status bit 2 (not used) | -- | ↑ | 8 | Status bit 4 (not used) | - |
| | 3 | System Ready | 14 | | 9 | } Not used | |
| | 4 | Status line 0 (not used) | - | | thru | | - |
| | 5 | Reset (not used) | -- | ↓ | 15 | | |
| | 6 | Drive Right | 16 | P5 | 16 | Data bit 01 | 2 |
| | 7 | Drive Left | 17 | P6 | 1 | } Not used | |
| | 8 | Control line 0 (not used) | -- | ↑ | thru | | |
| | 9 | Ground (not used) | -- | | 5 | | |
| | 10 | Drive Left return | 24 | | 6 | Data bit 06 | 7 |
| | 11 | Drive Right return | 11 | | 7 | Data bit 04 | 5 |
| | 12 | Ground (not used) | -- | | 8 | Data bit 02 | 3 |
| | 13 | Ground (not used) | -- | | 9 | Data bit 03 | 4 |
| | 14 | System Ready return | 13 | | 10 | Data bit 05 | 6 |
| | 15 | Ground (not used) | -- | | 11 | Data bit 07 | 8 |
| P4 | 16 | Data Ready return | 12 | | 12 | } Not used | |
| P5 | 1 | Data bit 00 | 1 | | thru | | -- |
| ↑ | 2 | } Not used | | | 15 | | |
| ↓ | thru | | -- | P6 | 16 | Ground (mode select) | 10 |
| P5 | 6 | | | | | | |

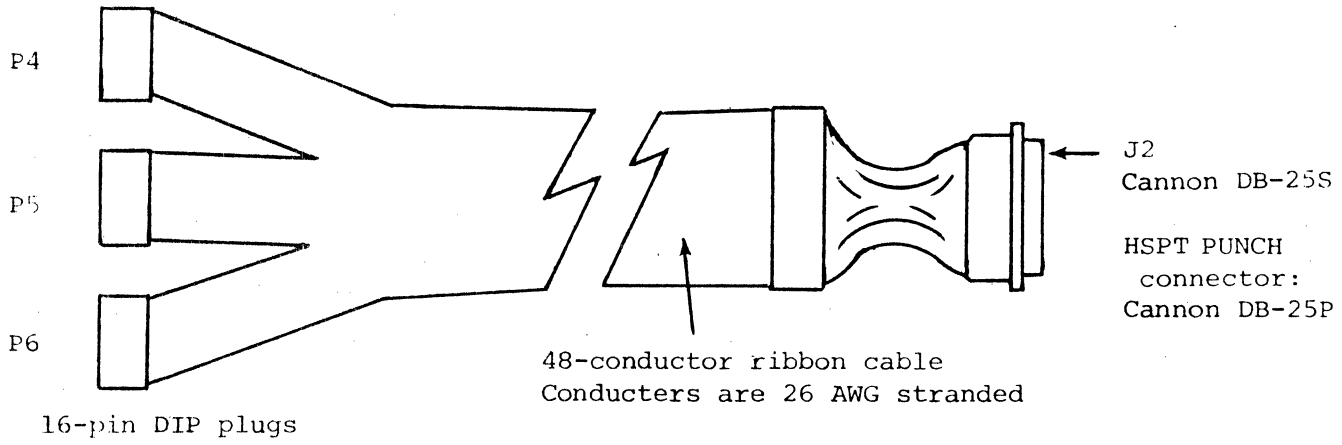



Figure 3-18. Cable Description - HSPT Reader

### 3.3.3.8 Programming Example

The Assembler Language statements shown in Table 3-3 demonstrate one method for using the Distributed I/O System. The technique is based on the programming information in Section 2.

The demonstration code falls into three distinct parts: Interrupt Locations, I/O Initiation and End-of-Block Service.

1. Interrupt Locations

   The first part of the code defines and fills the standard locations associated with IOD channel to which the device's PicoProcessor is connected. There are six locations involved of which two -- the Byte Count and the Buffer Location -- are dynamic for each I/O Initiation, while the remaining words can be fixed at program load time.

2. I/O Initiation

   The second part of the code accomplishes the transfer of one physical record each time the user's main-line program executes this instruction:

   ```
   CALL PREAD
   ```

   It is assumed that the calling program has previously set up these two words:

   | | |
   |---|---|
   | BYTCNT | Number of bytes to be transferred |
   | BUFADD | Word address of record buffer |

   The demonstration code converts this information into the form required by an Automatic I/O instruction: negative byte count and byte address of the buffer minus one. Once these computations have been stored into the interrupt locations, the PicoProcessor is sent a command specifying:

   ```
   Begin at Branch Address : 1
   ```

   Notice that the Device Address used in this sequence is coded with the same Device Address used in the interrupt location, plus one. The assembled object code will indicate the transfer of a command, rather than of data.

   No attempt is made in the demonstration code to overlap record transfer with CPU activity. Instead, a dead loop is issued -- JMP $ -- until the PicoProcessor interrupts not to the Data Service interrupt location, but to the End-of-Block location. At this point control passes to the subroutine labelled EOB.

3. End-of-Block Service

   The End-of-Block subroutine could perform error analysis, retry, etc. The demonstration code simply obtains device status from the PicoProcessor and passes status back to the main line program through the A register.

Table 3-3. Programming Example

```
PRINT   EQU     :FB         STANDARD PAPER TAPE READER INTERRUPT
PRDEVA  EQU     :FA         STANDARD PAPERTAPE READER DEVICE ADDRESS
*
        ABS     PRINT       INTERRUPT FOR DATA LOCATION
        AIB     PRDEVA      AUTO IN BYTE
        DATA    $-$         TO BE FILLED WITH COUNT
        DATA    $-$         TO BE FILLED WITH BUFFER ADDRESS-1
        DATA    4
*               INTERRUPT FOR END-OF-BLOCK
        JST     *+1         CALL END-OF-BLOCK ROUTINE
        DATA    EOB
*
PREAD   ENT
        LDA     BYTCNT      # OF BYTES IN RECORD
        NAR                 AIB INSTRUCTION NEEDS NEGATIVE
        STA     PRINT+1     PUT IN AIB INSTRUCTION
        LDA     BUFADD      ADDRESS (WORD) OF BUFFER
        LLA     1           AIB INSTRUCTION NEEDS BYTE ADDRESS
        SAI     1           STARTS AT -1
        STA     PRINT+2     PUT IN AIB INSTRUCTION
        LDA     =:M21B      WORD TO START PICOPROCESSOR
        OTA     PRDEVA+1    SEND TO PICO
        JMP     $           WAIT FOR END-OF-BLOCK
*
EOB     ENT     END-OF-BLOCK INTERRUPT SUBROUTINE
        INA     PRDEVA+1    INPUT STATUS
        RTN     PREAD       RETURN TO CALLER WITH
*                           STATUS IN A REGISTER
```

### 3.3.4 High-Speed Paper Tape Punch

#### 3.3.4.1 Description

Intelligent Cable 14631-04 controls the transfer of eight-bit positive-true parallel data from the computer to a Remex Model RAB6375 High-Speed Paper Tape Punch. The cable's PicoProcessor accepts the IOD output command, interrupts the CPU for data, and then issues a Punch command to perforate the appropriate track of the tape when a "1" is to be entered. When all data has been transferred or when an error is detected, the PicoProcessor generates an End-of-Block interrupt to terminate the operation.

NOTE

This PicoProcessor is designed to operate with a Remex Model 6375 Reader/Perforator having a specific part number of RAB6375BA1/661/551/U000. The "551" indicates an option to the standard "550" model. When ordering a new Reader/Perforator from Remex, order by the above part number. If the user has an existing "550" model, it can be modified as explained in 3.3.4.8.

#### 3.3.4.2 Physical Details

Cable Lengths:
IOD to PicoProcessor, 4 feet
PicoProcessor to Punch, 1-1/2 feet

Standard Channel Number, 6 (Device Address field = :FC)

Standard Data Service Interrupt Address, :F0

Standard End-of-Block Interrupt Address, :F4

#### 3.3.4.3 HSPT Punch Status Word

To control the transfer of data to the Paper Tape Punch, the PicoProcessor sequences through a series of operations based, in part, on the state of individual bits of the Paper Tape Punch Status Word. The Status Word, shown in figure 3-19, indicates certain operational or error conditions in the Paper Tape Punch. When the PicoProcessor receives

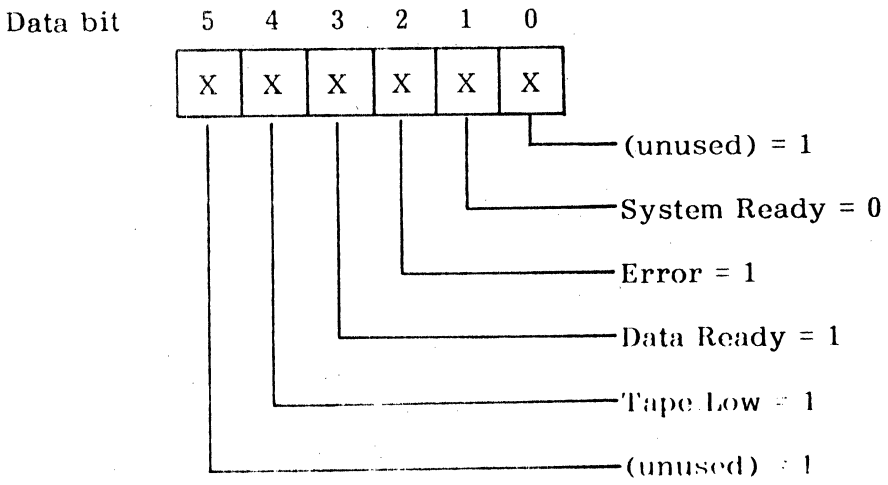



Figure 3-19. Paper Tape Punch Status Word

an input instruction requesting device status, it immediately transfers the entire Status Word to the CPU on bits 0 thru 5 of the Data bus. Input status can be requested at any time, but it is usually done after an End-of-Block interrupt to determine the reason for termination. Descriptions of the individual status bits are given under 3.3.4.5.

### 3.3.4.4 Operating Sequence

The PicoProcessor has 16 unique sequence addresses (:0 thru :F). When the PicoProcessor receives a Command Word with the Begin bit set to "1", it immediately begins operation at the sequence address specified by the Branch Address field of the Command Word. Standard software enters the sequence at the following sequence addresses:

| Sequence Address | Operation |
|---|---|
| :1 | Check Tape Low status; Start Punch Operation. |
| :2 | Start Punch Operation (skip Tape Low status check). |

Details of the PicoProcessor operating sequence are shown in the flow chart (figure 3-20). Interface lines are described in 3.3.4.5. Following is a sequence description with each segment of the operation identified by name and sequence address (:0 thru :F). The yes/no decisions refer to the true/false state of a line, regardless of actual logic level.

IDLE (:0)

The PicoProcessor is initially in the Idle state as a result of a Reset command or because of the completion of an end-of-block interrupt. A Begin command with a starting address of :1 or :2 takes the PicoProcessor out of the Idle state and into the Start sequence.

START (:1)

The PicoProcessor begins the Start at :1 sequence by checking device status. Tape Low is checked first. If this line is true, the PicoProcessor generates an end-of-block interrupt to terminate the operation.

START (:2)

A Start at :2 sequence begins at this point if it is desired to skip the Tape Low status check. First, error status is checked. The Error status line is true when the tape is broken or loose or when the chad box is full. On a true response, the PicoProcessor generates an end-of-block interrupt. Otherwise, System Ready status is checked next. A false response indicates that power is not applied to the Punch or that the internal voltages of the Punch have not stabilized. If false, the PicoProcessor generates an end-of-block interrupt. If System Ready is true, the PicoProcessor advances to Data Interrupt.

DATA INTERRUPT (:A)

The PicoProcessor generates a data service interrupt causing the automatic output instruction at the interrupt location to be executed. The CPU transfers data to the PicoProcessor and increments the byte count and memory buffer pointer. A line from the CPU is activated if the byte count increments to zero to signal the PicoProcessor (via the IOD) that all data has been transferred (end-of-block).

While the PicoProcessor checks if the byte count = 0, the Punch command line is activated. If BC=0, the PicoProcessor sustains the Punch command while waiting for the Data Ready line to go false and then, still sustaining Punch, checks System Ready. If System Ready is false, an end-of-block interrupt is generated. Otherwise, data is transferred to the HSPT Punch when Data Ready goes true. The PicoProcessor then generates an end-of-block interrupt.

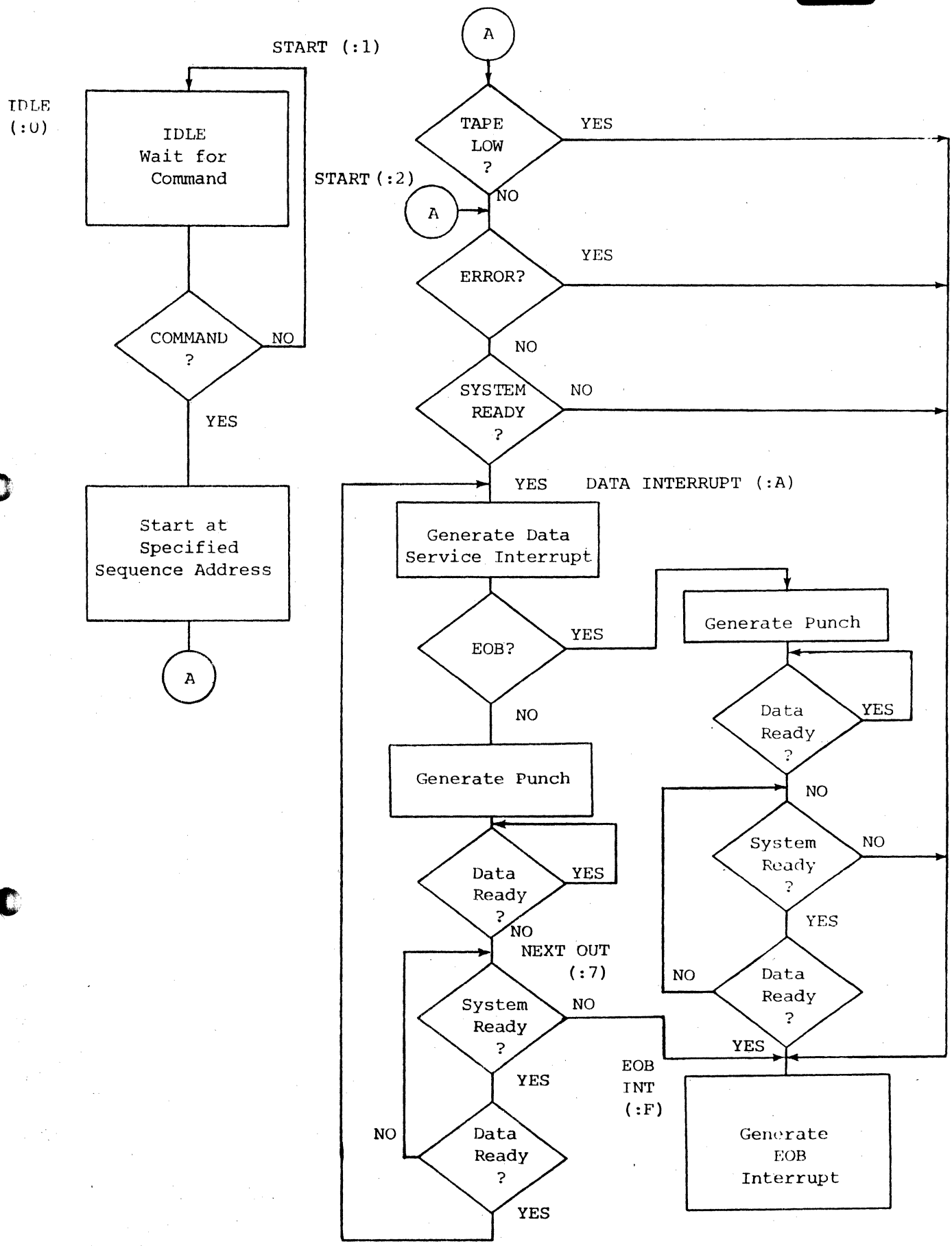



Figure 3-20. Firmware Sequence - HSPT Punch Intelligent Cable

If the byte count is not zero, the PicoProcessor enters the Next Out sequence.

NEXT OUT (:7)

The PicoProcessor sustains the Punch command and checks the Data Ready status line. It waits for the Data Ready line to go false, and then checks System Ready, still sustaining the Punch command. If power is still applied to the HSPT (System Ready true) data is transferred to the Punch when Data Ready goes true. The PicoProcessor then repeats the Data Interrupt sequence until byte count = 0 is received.

EOB INTERRUPT (:F)

The end-of-block interrupt is generated by the PicoProcessor when the last data byte has been transferred or on detection of a status error at any point in the sequence. When the end-of-block interrupt operation is completed, the CPU commands the PicoProcessor (via the IOD) to return to the Idle state and wait for the next Begin command.

3.3.4.5 Interface Description

Interface lines between PicoProcessor and Paper Tape Punch (figure 3-21) include eight data lines, one control line to the Punch and four status lines from the Punch.

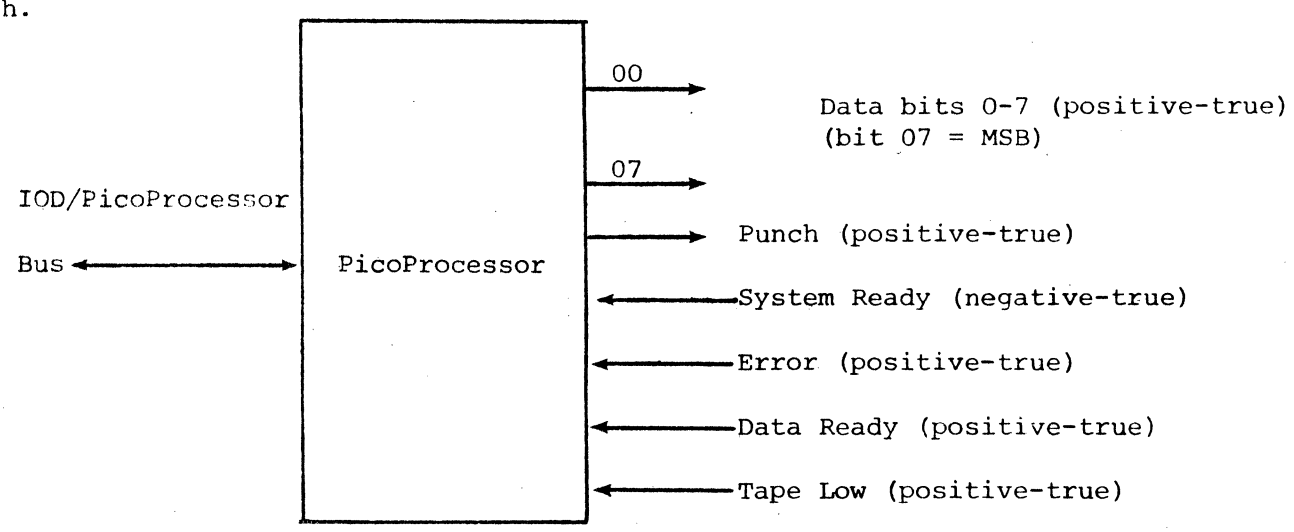



Figure 3-21. Paper Tape Punch Interface

3.3.4.5.1 Control Lines (to Paper Tape Punch)

1. Punch. This line, when true (+5V), starts the tape moving and initiates punching.

2. Reset. This negative-true line is driven by the CPU RESET switch or under software control. Under software control, it is a 250 ns pulse. This line is not used by the Paper Tape Punch.

3.3.4.5.2 Status Lines (from Paper Tape Punch).

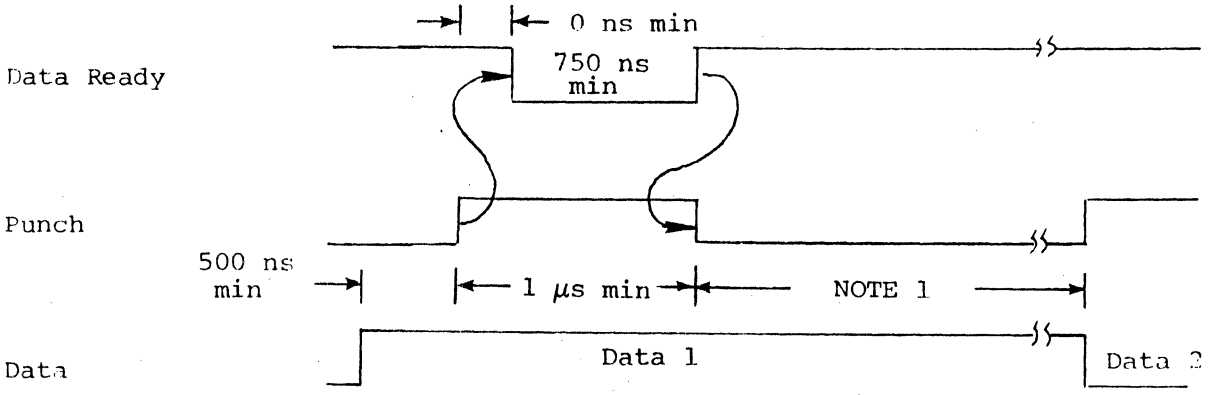
1. System Ready. This line is true (0 volts) when power is applied to the Punch and its internal voltages have stabilized. The PicoProcessor checks the status of this line before every data transfer.

2. Error. This line is true (+5 volts) when the Punch is not in the Run mode, the paper tape is broken, loose or tight or when the chad drawer is full. This line is checked only once for every block of data transferred.

3. Data Ready. This line is true (+5 volts) when the Punch is ready to accept a Punch command. It is false during the "advance and punch" cycle (approximately 13 msec after a Punch command).

4. Tape Low. This line, when true, (+5 volts) indicates that the tape supply is nearly exhausted. It is for information only. Operation of the punch is not affected. This line is checked once for every block of data transferred, if started at sequence address :1.

Figure 3-22 shows the interface timing. For additional details, see the Remex Paper Tape Reader/Punch instruction manual.




NOTE 1: Minimum time equal to the instruction time of the Auto I/O byte instruction plus 4 sec. For instruction time, see the Appendix of the appropriate Computer Handbook.

Figure 3-22. Interface Timing - Paper Tape Punch Intelligent Cable

3.3.4.5.3 Data Lines. The eight data lines to the Paper Tape Punch (0 through 7) are positive-true with bit 7 the most-significant. Data is output to the Punch in standard eight-bit ASCII characters.

3.3.4.6 Strapping Requirements

The Sequence Select factory-installed jumper is installed across pins 5 and 12 of J3 in the PICOPROCESSOR for the Paper Tape Punch. The location of J3 is shown in figure 3-23.

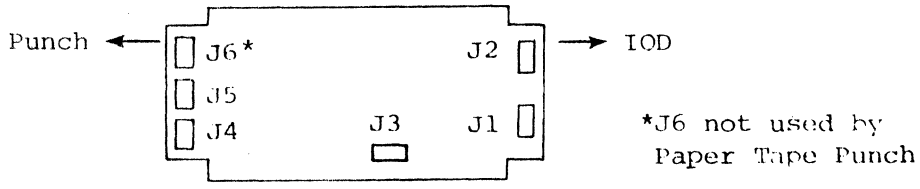



Figure 3-23. Connector Location - HSPT Punch PicoProcessor

3.3.4.7 Device Cable Description

The device cable is 18-inches long and terminated on the PicoProcessor end with two 16-pin DIP plugs (P4 and P5). The HSPT Punch end of the cable is terminated with a 25-pin connector that mates with connector J1 on the Paper Tape Punch.

Figure 3-24 lists all interface lines in the device cable and identifies the connectors used. Location of mating connectors on the PicoProcessor is shown in figure 3-23.

3.3.4.8 HSPT Punch Modification

The Intelligent Cable for the HSPT Punch is designed to operate with a Remex Model RAB6375 having the following part number: RAB6375BA1/661/551/U000. This model can be ordered from Remex.

Models, having a part number of RABxx75Bxx/6xx/55*0/U000, can be modified to the required configuration. Folowing are instructions for modifying a model to the "551" configuration.

General Procedure

Because of the great number of previous models of Remex Reader/Perforators and stand alone Perforators, it is not reasonable to give a detailed modification procedure for every possible model. The following general procedure should be sufficient to allow a user to modify his own system.

1. Find the Remex Model Number Code breakdown for your Perforator or combination Reader/Perforator in the Remex Technical Manual that was provided with your unit. This breakdown should be in Section 1 of the manual.

   Locate the digit which specified "Other Perforator Circuit Options" shown by asterisk in the number at the top of this page.

   0 = Standard
   1 = Tape Handling Error condition does not inhibit Punch Ready output

2. Using the Model Code as a guide, examine the "Other Perforator Circuit Options" digit in the Remex Model Number tag on your unit.

   If the unit has a "1" in this position, no modification to the unit is necessary. If the digit is a "0", continue with this modification procedure.

3. Find the assembly drawing for the Perforator Control Logic board in Section 7.

   Usually, an arrow will point to a jumper which has a note reading something like:

   | | |
   |---|---|
   | Add Jumper | "Jumper Added |
   | For XXXXXX-X0 | For -1 Assy." |
   | (No options) | |

   Remove this jumper.

   If no jumper is identified on the assembly drawing, go to the schematic for the Perforator Control Logic board and locate the jumper connecting the Tape Handling Error logic with the Punch Ready logic. Note the letters at the end points of the jumper and remove the corresponding jumper on the PC board.

4. Attach a label next to the old Remex model number on the unit showing the new model number. The new model number is the same as the old model number except that a "1" will appear in the "Other Perforator Circuit Options" position.

Specific Procedure for Converting from RAB6375BA1/661/55*0/U000 to RAB6375BA1/661/551/U000

1. Check the "Other Perforator Circuit Options" bit of the Remex model number shown by asterisk (above). The bit position can be found by referring to Page 1-10 of Remex Technical Manual RSM-304Y-1.

If the bit is a "1", the option is already installed and no modification is necessary. If the bit is a "0", continue with this modification procedure.

2. For units with serial numbers 45261 and higher, refer to page 7-32 of the Technical Manual and remove the jumper referred to in the note:

   "Jumper Added
   For -1 Assy."

   on the Perforator Logic and Driver Card 111031. Change the last digit in the card assembly dash number from 1 to 2. Proceed to Step 3.

3. Attach a new label next to the old Remex part number label on the unit with the new Remex part number. The new part number is the same as the old part number except that a "1' will appear in the bit position for "Other Perforator Options".

| PicoProcessor Conn | PicoProcessor Pin | Description | Reader/ Punch J1 pin | PicoProcessor Conn | PicoProcessor Pin | Description | Reader/ Punch J1 pin |
|---|---|---|---|---|---|---|---|
| P4 | 1 | Data Ready | 12 | P5 | 1 | Not used | - |
| | 2 | Error | 20 | | 2 | Ground (not used) | - |
| | 3 | System Ready | 13 | | 3 | Data bit 06 | 7 |
| | 4 | Status bit 0 (not used) | -- | | 4 | Data bit 04 | 5 |
| | 5 | Reset (not used) | -- | | 5 | Data bit 02 | 3 |
| | 6 | Punch | 11 | | 6 | Data bit 00 | 1 |
| | 7 | Control line 1 (not used) | -- | | 7 | Status bit 5 (not used) | - |
| | 8 | Control line 0 (not used) | -- | | 8 | Tape Low | 21 |
| | 9 | Ground (not used) | -- | | 9 | Ground (not used) | |
| | 10 | Ground (not used) | -- | | 10 | Ground (not used) | |
| | 11 | Ground | 25 | | 11 | Data bit 01 | 2 |
| | 12 | Ground (not used) | -- | | 12 | Data bit 03 | 4 |
| | 13 | Ground (not used) | -- | | 13 | Data bit 05 | 6 |
| | 14 | Ground | 23 | | 14 | Data bit 07 | 8 |
| | 15 | Ground | 18 | | 15 | Ground (not used) | - |
| P4 | 16 | Ground (not used) | -- | P5 | 16 | Not used (input data) | - |

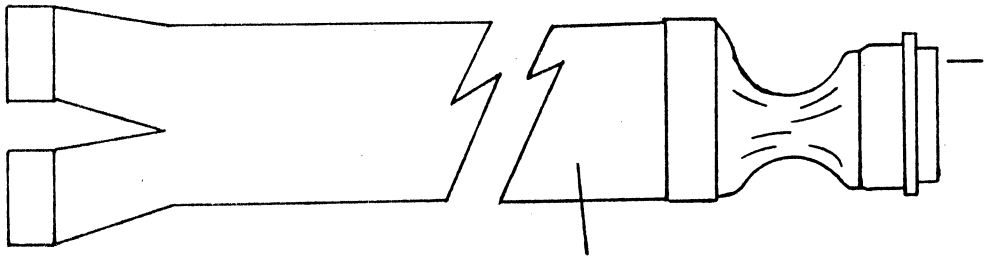



Figure 3-24. Cable Description - HSPT Punch

3.3.4.9 Programming Example

The Assembler Language statements shown in Table 3-4 demonstrate one method for using the Distributed I/O System. The technique is based on the programming information in Section 2.

The demonstration code falls into three distinct parts: Interrupt Locations, I/O Initiation and End-of-Block Service.

1. Interrupt Locations

The first part of the code defines and fills the standard locations associated with IOD channel to which the device's PicoProcessor is connected. There are six locations involved of which two -- the Byte Count and the Buffer Location -- are dynamic for each I/O initiation, while the remaining words can be fixed at program load time.

2. I/O Initiation

The second part of the code accomplishes the transfer of one physical record each time the user's main-line program executes this instruction:

CALL PUNCH

It is assumed that the calling program has previously set up these two words:

| | |
|---|---|
| BYTCNT | Number of bytes to be transferred |
| BUFADD | Word address of record buffer |

The demonstration code converts this information into the form required by an Automatic I/O instruction: negative byte count and byte address of the buffer minus one. Once these computations have been stored into the interrupt locations, the PicoProcessor is sent a command specifying:

Begin at Branch Address : 1

Notice that the Device Address used in this sequence is coded with the same Device Address used in the interrupt location, plus one. The assembled object code will indicate the transfer of a command, rather than of data.

No attempt is made in the demonstration code to overlap record transfer with CPU activity. Instead, a dead loop is issued -- JMP $ -- until the PicoProcessor interrupts not to the Data Service interrupt location, but to the End-of-Block location. At this point control passes to the subroutine labelled EOB.

3. End-of-Block Service

The End-of-Block subroutine could perform error analysis, retry, etc. The demonstration code simply obtains device status from the PicoProcessor and passes status back to the main-line program through the A register.

Table 3-4. Programming Example

```
PPINT   EQU     :F0             STANDARD PUNCH INTERRUPT
PPDEVA  EQU     :FC             STANDARD PUNCH DEVICE ADDRESS
        ABS     PPINT           INTERRUPT FOR DATA LOCATION
        AOB     PPDEVA          AUTO OUTPUT BYTE
        DATA    $-$             TO BE FILLED WITH BYTE COUNT
        DATA    $-$             TO BE FILLED WITH BUFFER ADDRESS -1
        DATA    0
*                               INTERRUPT FOR END OF BLOCK
        JST     *$+1            CALL END-OF-BLOCK ROUTINE
        DATA    EOB
PUNCH   ENT                     ENTRY POINT FOR PUNCH DRIVER
        LDA     BYTCNT          BYTE COUNT FOR MESSAGE
        NAR                     AOB INSTRUCTION NEEDS NEGATIVE
        STA     PPINT+1         PUT IN AOB INSTRUCTION
        LDA     BUFADD          ADDRESS (WORD) OF BUFFER
        LLA     1               AOB INSTRUCTION NEEDS BYTE ADDRESS
        SAI     1               STARTS AT -1
        STA     PPINT+2         PUT IN AOB INSTRUCTION
        LDA     =:0210          WORD TO START PICOPROCESSOR
        OTA     PPDEVA+1        SEND COMMAND TO PICO
        JMP     $               WAIT FOR END-OF-BLOCK
EOB     ENT                     END-OF-BLOCK INTERRUPT SUBROUTINE
        INA     PPDEVA+1        INPUT STATUS
        XTN     PUNCH           RETURN TO CALLER WITH
*                               STATUS IN A REGISTER
```

### 3.3.5 Teletype

#### 3.3.5.1 Description

Intelligent cable 14632-01 provides a current loop interface between a modified ASR-33 Teletype and an LSI Series computer. This Intelligent Cable provides a Teletype motor on/off control and a Reader control for the Paper Tape Reader. Therefore, the Teletype should be modified for improved performance of certain remote control functions as described in Computer Automation document 91-22215. The cable's PicoProcessor performs all of the data and control signal conversion (serial-to-parallel and parallel-to-serial) required to control Teletype operation and data transfer to and from the device.

The Teletype PicoProcessor controls Teletype operation in half-duplex mode in which data is transmitted to or from the Teletype, but not simultaneously. Use of the "Echo Incoming Data" feature causes input from the device to be automatically "echoed" back to the Teletype for printback without the need for separate software instructions to obtain a printback.

The selection of an input or output operation and the selection of the Echo feature are made by setting the appropriate mode control bits of the Command Word. Additional bits of the Command Word are set to select character detection and parity bit standardization.

The Character Detection feature provides automatic detection of the 7-bit ASCII Carriage Return character (:0D). The PicoProcessor generates an End-of-Block interrupt when Carriage Return is received to terminate the input operation even though the input buffer (specified by the Auto I/O instruction) has not been filled (byte count ≠ 0).

When the Parity Bit Standardization feature is enabled, the most-significant bit of all incoming data characters (bit 7) is set to the same value independent of its actual value. In this way, all characters can be examined in the same operation, whether they use off or even parity.

#### 3.3.5.2 Physical Details

Cable Lengths:
PicoProcessor to IOD, 4 feet
PicoProcessor to Teletype, 12 feet

Standard Channel Number, 4 (device address field=:F8)
Standard Data Service Interrupt Address, :E0
Standard End-of-Block Interrupt Address, :E4
Interface Logic (see 3.3.5.6)

#### 3.3.5.3 Teletype Status Word

To control the transfer of data to and from the Teletype, the PicoProcessor sequences through a series of operations based, in part, on the state of individual bits of the Teletype Status Word. The Status Word, shown in figure 3-25, indicates certain operational and error conditions in the Teletype. When the PicoProcessor receives an input instruction requesting device status, it immediately transfers the entire Status Word to the CPU on bits 0 through 5 of the data bus. Input status can be requested at any time, but is usually requested after an End-of-Block interrupt to determine the reason for termination.

An EOB interrupt caused by an Overrun or Parity Error will not be indicated by the status word. If all status bits are in the proper state, the byte count at the appropriate memory location must be checked (for byte count = 0). If Character Detection is selected, the appropriate memory location must be checked for receipt of a Carriage Return character. If the byte count does not equal zero and if Character Detection is selected and no Carriage Return has been received, an Overrun or Parity Error exists.

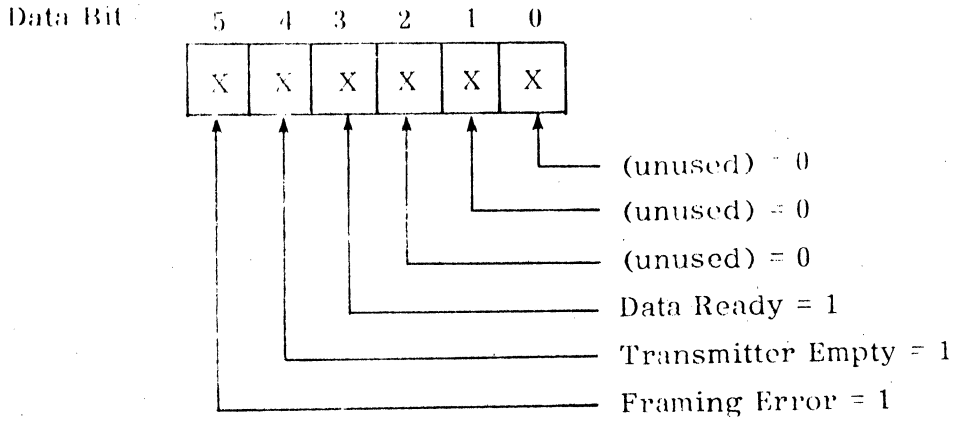



Figure 3-25. Teletype Status Word

Following is a definition of each status bit.

Framing Error. This status line is true to indicate that a valid stop bit was not present in the received character or the break key was pushed. This line is reset prior to each transfer block and updated for each character transferred.

Data Ready. This line is true during an input operation to indicate that an entire character has been received by the PicoProcessor Receiver. Data is available to be transferred to the CPU. This line is reset after each data character has been transferred to the CPU.

Transmitter Empty. This line is true during an output operation to indicate that the PicoProcessor Transmitter (buffer) has serially transmitted an entire character to the Teletype. Another character can now be transferred from the CPU. It is reset prior to each block transfer.

3.3.5.4 Mode Bit Significance

Certain PicoProcessor and device operations are selected by the mode bits (bits 0-3) of the Command Word. Additional bits of the Command Word are set to select character detection and parity bit standardization.

Bit 0 (Teletype Motor Off). The state of this bit controls operation of the Teletype motor. When this bit is a "0", the motor is turned on; when a "1", the motor is turned off. This bit is reset to "motor on" condition. After the motor has been off, it requires 600 ms to attain operating speed when it is turned on. This delay must be accomplished by software.

Bit 1 (Select Output). This bit selects a data output operation. It is set to "1" to perform a data output transfer. A "0" level selects an input data transfer. The bit

is used internally to control the PicoProcessor operating sequence to perform either an input or output operation. The bit is reset to the input sequence.

Bit 2 (Enable Teletype Reader). This bit, when true (=1), enables operation of the Teletype Paper Tape Reader. An input operation will then input data from the Teletype Reader. When this bit is false (0), an input operation inputs data from the Teletype keyboard. The bit is reset to the keyboard input mode.

Bit 3 (Echo Incoming Data). A "1" in this bit selects the Echo mode of operation in which the serial input data is returned or "echoed" back to the device as it is received by the PicoProcessor. It is typically used to obtain a printback of data input to the PicoProcessor from the Teletype keyboard, but will echo data from the Reader as well. The bit is reset to the non-echo mode. This bit is not significant in an output operation.

3.3.5.5 Operating Sequence

The PicoProcessor has 16 unique sequence addresses (:0 through :F). When it receives a Command Word with the Begin bit set to "1", it immediately begins operation at the sequence address specified by the Branch Address field of the Command Word. Standard software enters the sequence at one of the following sequence addresses. All mode bits are zero unless otherwise specified as "1". Teletype motor is "on" for each operation.

| Sequence Address | Operation |
|---|---|
| :1 | Start Input |
| :1 (mode bit 2 = 1) | Start Input (Reader Enabled) |
| :1 (mode bit 3 = 1) | Start Input in Echo Mode |
| :1 (mode bits 2 and 3 = 1) | Start Input (Reader enabled, Echo Mode) |
| :1 (mode bit 1 = 1) | Start Output |

3.3.5.5.1 Output Operation. Details of the TTY PicoProcessor operating sequence for an output operation are shown in the flow chart (figure 3-26). The operating sequence (firmware) controls the transfer of data from the CPU to the PicoProcessor. The actual data transfer to the Teletype is performed by a Serial Transmitter in the PicoProcessor which also converts the data from parallel-to-serial and provides buffering. Operation of the transmitter does not appear on the flow chart, but firmware timing and generation of interrupts is done on the basis of signals from the transmitter.

Following is a description of the flow chart with each segment of the sequence identified by name and sequence address.

IDLE (:0)

The PicoProcessor is initially in the Idle state with the Teletype motor turned on as a result of a Reset command or because of the generation of an End-of-Block interrupt. A Begin command starts operation at the sequence address specified by the Branch Address field of the Command Word.

START (:1)

PicoProcessor operation starts with the generation of an internal Reset pulse to reset

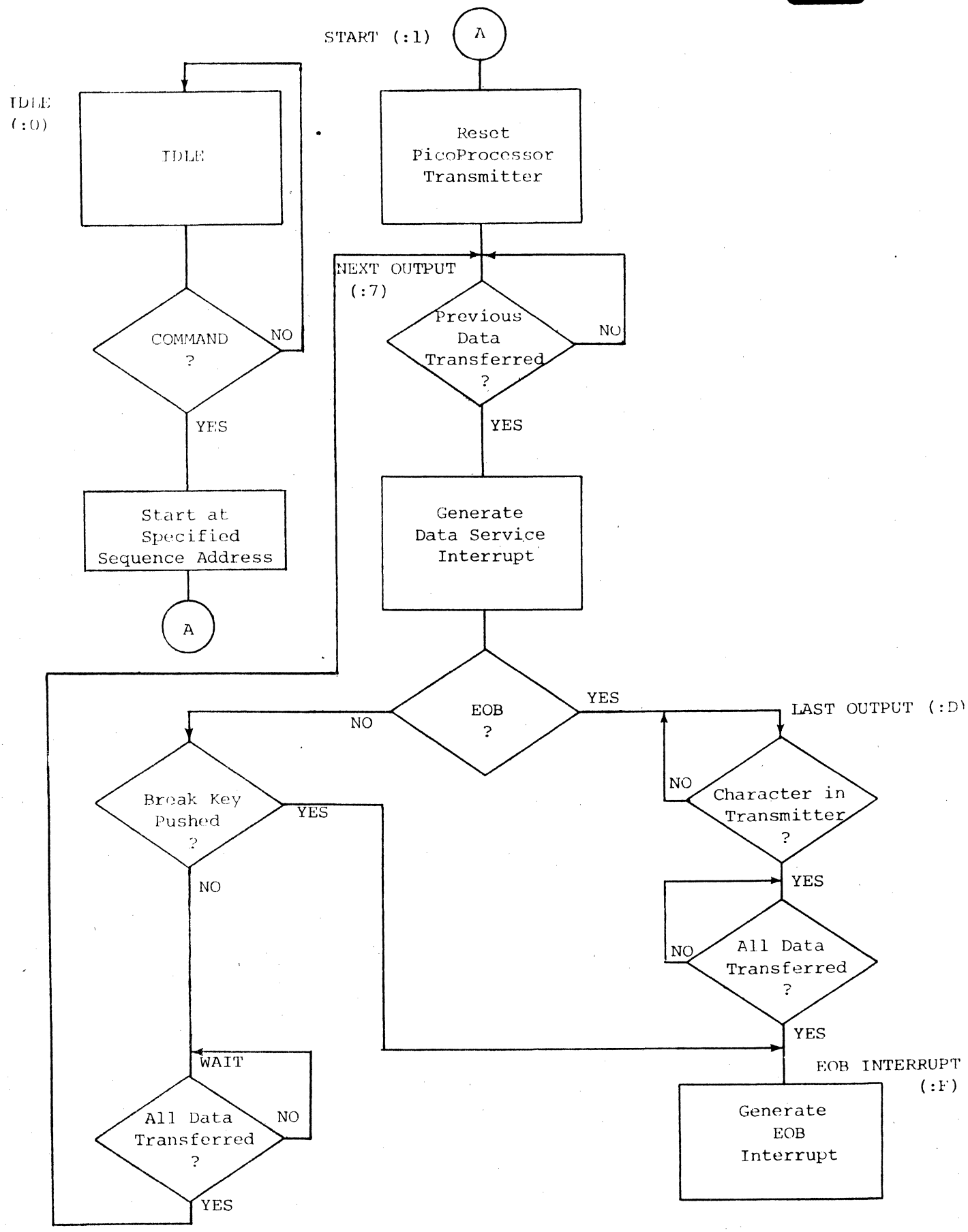



Figure 3-26. Firmware Sequence - TTY Output

NEXT OUTPUT (:7)

The PicoProcessor monitors a status line from the transmitter buffer (Transmitter Empty) which is true when all data (previously transferred from the CPU) has been serially shifted out to the Teletype (previous data transferred). The PicoProcessor waits for this line to go true, then generates a Data Service interrupt.

The Auto I/O (output) instruction at the interrupt location is executed and one character of data is transferred from the CPU to the PicoProcessor. The byte count and memory address pointer are incremented. If the data was not the last to be transferred, byte count = 0. The PicoProcessor then checks for a Framing Error and generates an End of Block interrupt if an error exists.

The Transmitter Empty line is again monitored. The PicoProcessor waits for this line to go false indicating that the data transferred as a result of the Data Service interrupt is now in the transmitter. The PicoProcessor then repeats the Next Output Sequence where it waits for the transmitter to signal that the entire character has been serially transferred to the Teletype. It then generates another Data Service interrupt to output another character from the CPU.

LAST OUTPUT (:D)

When the last data character has been transferred to the PicoProcessor, the PicoProcessor monitors the Transmitter Empty status line. It waits, first, for the line to be false indicating that the character from the CPU is now in the transmitter. It then waits for the line to go true indicating that the entire character has been serially shifted out to the Teletype (all data transferred).

EOB INTERRUPT (:F)

After the entire character has been shifted out of the transmitter at the end of the Last Output, the PicoProcessor generates an End-of-Block interrupt. The interrupt is also generated when a Framing Error is detected. Upon service of the interrupt, the PicoProcessor returns to the Idle state.

3.3.5.5.2 Input Operation. Details of the TTY PicoProcessor operating sequence for an input operation is shown in the flow chart (figure 3-27). The firmware sequence controls the transfer of data from the PicoProcessor to the CPU. The actual transfer of data from the Teletype to the PicoProcessor is performed by a Serial Receiver in the PicoProcessor which also converts the input data from serial to parallel. Operation of the Receiver does not appear in the flow chart. Following is a description of the flow chart with each segment of the sequence identified by name and sequence address.

IDLE (:0)

The PicoProcessor is initially in the Idle state with the Teletype motor on as a result of a Reset command or because of the generation of an End-of-Block interrupt. A Begin command starts operation at the sequence address specified by the Branch Address field of the Command Word.

START (:1)

PicoProcessor operation starts with the generation of an internal Reset pulse to clear the PicoProcessor's Receiver.

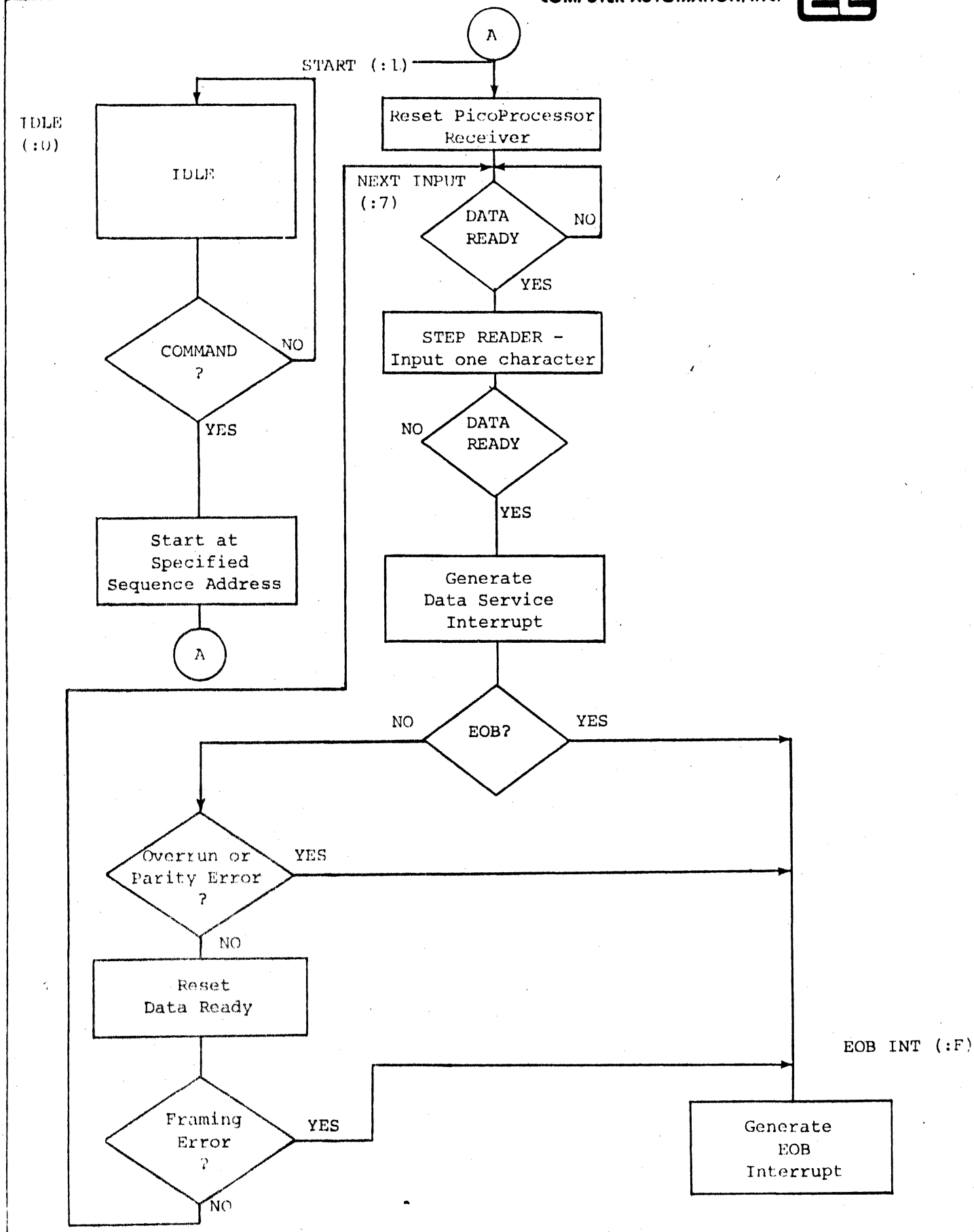



Figure 3-27. Firmware Sequence - TTY Input

NEXT INPUT (:7)

The PicoProcessor waits for the Receiver to be cleared. The PicoProcessor generates a signal to step the Teletype Reader one character position. Data is then input from the Teletype in serial form until the Receiver has input an entire character. This is indicated by the Data Ready status line going true. The PicoProcessor then generates a Data Service interrupt to transfer the character to the CPU.

If the character was not the last to be transferred, the input character is checked for an Overrun or Parity error, a Data Ready reset is generated and then Framing Error status is checked. If any error exists, an immediate End-of-Block interrupt is generated. Otherwise, the PicoProcessor returns to the Next Input sequence where it waits for the Receiver to clear and then inputs another character.

EOB INTERRUPT (:F)

If the Character was the last to be transferred, the PicoProcessor generates an End-of Block interrupt. It also generates an End-of-Block on an Overrun, Parity or Framing Error condition. Upon service of the interrupt, the PicoProcessor returns to the Idle state

3.3.5.6 Interface Description

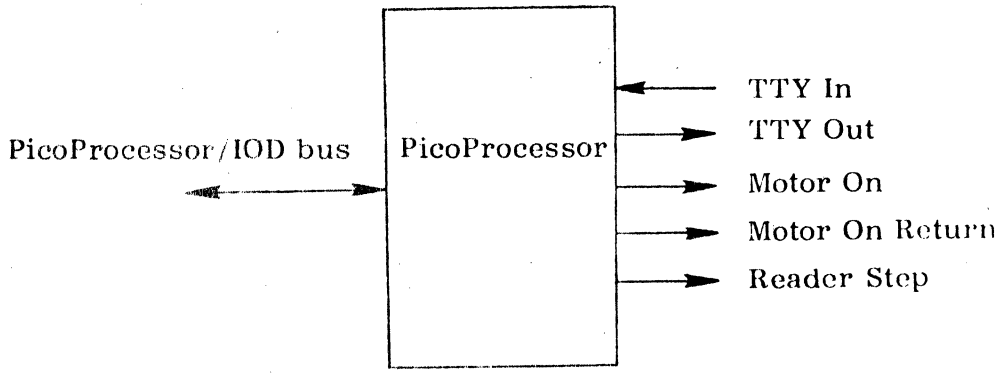


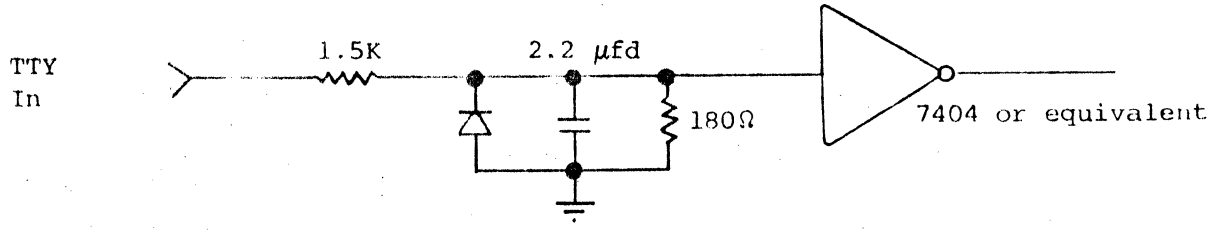

Figure 3-28. Teletype Interface Lines

Interface lines between the PicoProcessor and the Teletype (figure 3-28) include one serial data output line, one serial data input line, one control line to the Teletype motor and one control line to the Paper Tape Reader. Interface timing is shown in figure 3-30.

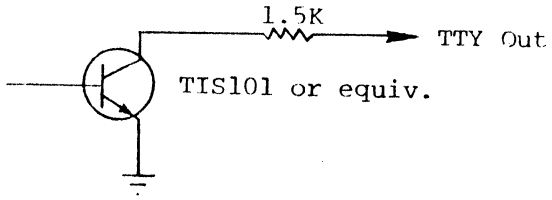
1. TTY In. This is the serial data input line from the Teletype. Its levels are

   mark = 20 ma current flow = 1
   space = absense of current = 0

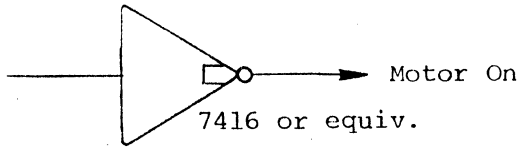
   The PicoProcessor serial data input circuit is shown below. The minimum required loop voltage supply for 20 ma operation is 30 volts (maximum 70 volts).



2. TTY Out. This is the serial data output line to the Teletype. The PicoProcessor serial data output circuit for driving a current loop device is a common-emitter current switch capable of sinking a minimum of 20 ma (shown below).




3. Motor On. This negative-true line is under the control of mode bit 0 (in the Command Word). It turns the Teletype motor on or off. The line uses TTL logic levels where true (OV) turns the motor on and false (+5V) turns the motor off (shown below).




4. Motor On Return. This line is a signal ground.

5. Step Reader. This line is pulsed under the control of the PicoProcessor and enabled by mode bit 2 (of the Command Word). It uses TTL logic levels where true (OV) steps the Reader.

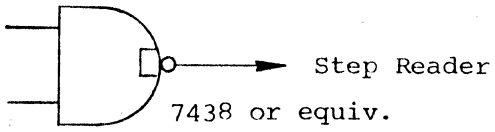



Figure 3-29 shows the detailed current waveform for the current loop intelligent cable. Interface timing is shown in figure 3-30.

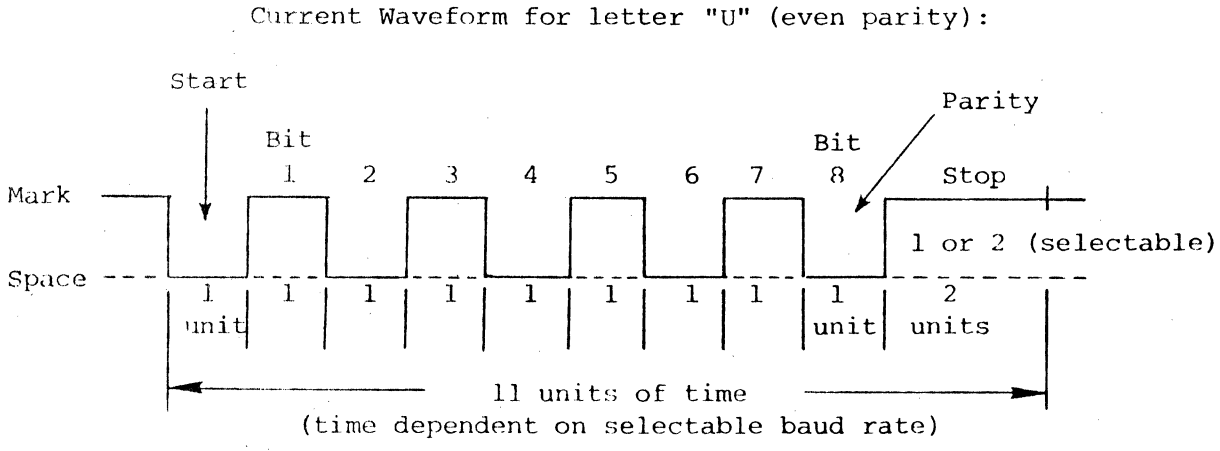
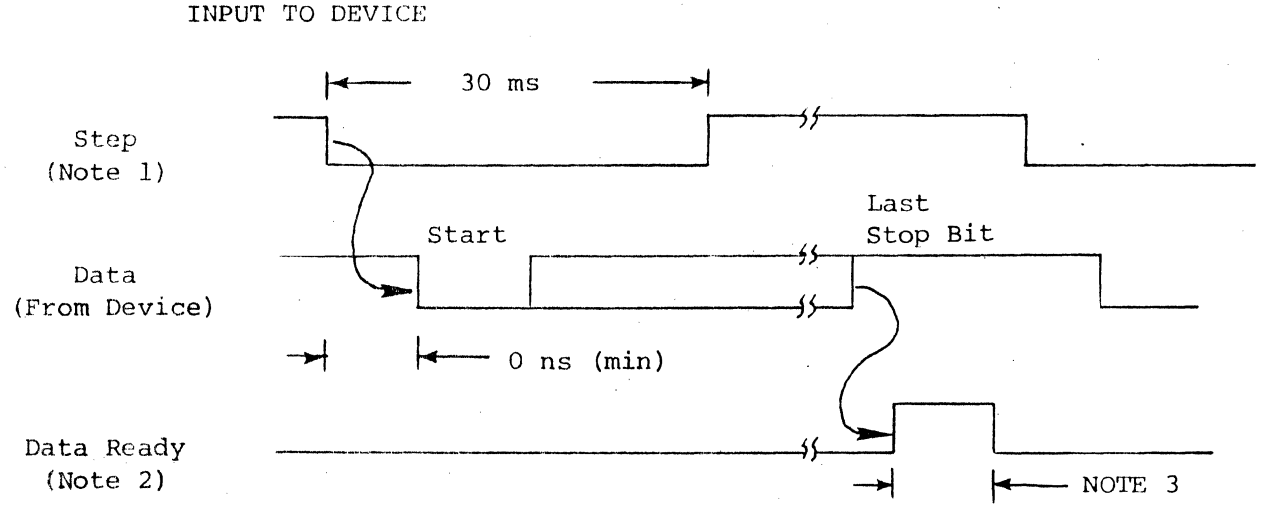


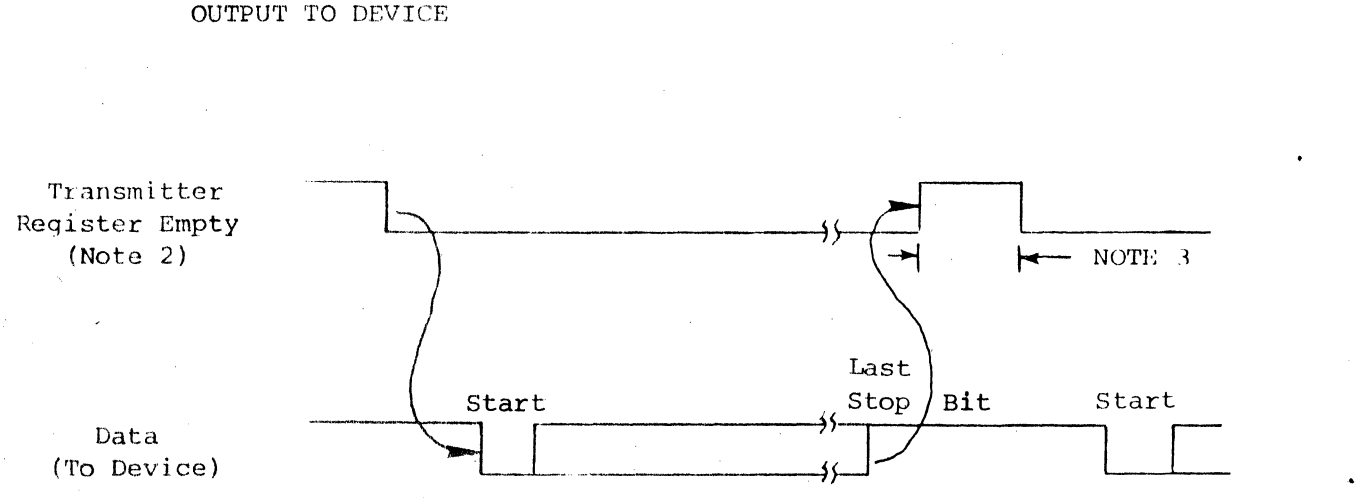

Figure 3-29. Current Waveform - Current Loop Intelligent Cable

3.3.5.7 Strapping Requirements - See figure 3-31.

3.3.5.8 Device Cable Description

The Teletype cable is 12-feet long and terminated on the PicoProcessor end with one 14 pin DIP plug (P4). The Teletype end of the cable is terminated with a 15-pin connector that mates with Teletype connector J2. Figure 3-32 lists all interface lines in the cable and identifies the connectors used. The location of the mating connector on the PicoProcessor is shown in figure 3-31.




NOTES:

1 - For Reader input only
2 - Status line available only to Programmer
3 - Minimum time equal to the instruction time of the Auto I/O byte instruction plus 4 sec. For instruction time, see the Appendix of the appropriate Computer Handbook.

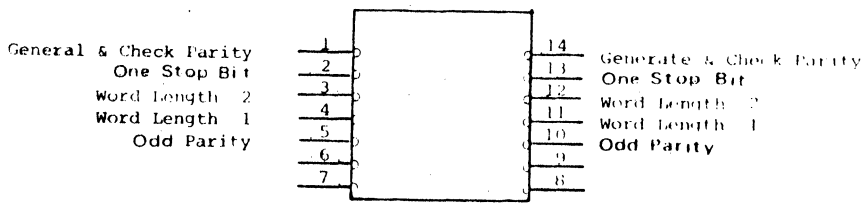
Figure 3-30. Interface Timing - Current Loop Intelligent Cable

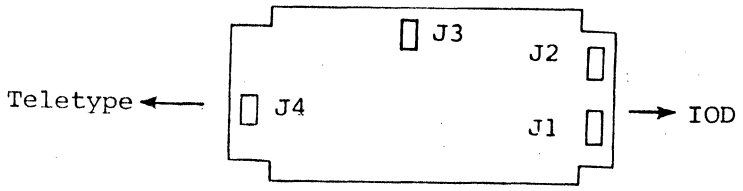
Word Length Selection

| Word Length Data Bits | Jumper 3 to 12 | Jumper 4 to 11 |
|---|---|---|
| 8 (Standard) | Open | Open |
| 7 | Open | In |
| 6 | In | Open |
| 5 | In | In |

Generate and check parity, Open = Disabled
In = 1 Bit
Odd/even parity, Open = even parity
In = Odd Parity

J3 Configuration




Install the required jumpers on the J3 header. Use Number 24 to 28 solid wire. Solder all connections. All jumpers go straight across the header.

Figure 3-31. Connector Locations - Teletype PicoProcessor

| From Conn | From Pin | Description | To J2 Pin | From Conn | From Pin | Description | To J2 Pin |
|---|---|---|---|---|---|---|---|
| P4 | 1 | Ground (not used) | -- | P4 | 11 | Motor On | 12 |
| ↑ | 2 | Ground (not used) | -- | ↑ | 12 | Step Reader | 15 |
| | 3 | Ground (not used) | -- | ↓ | 13 | Ground (not used) | -- |
| | 4 | Ground (not used) | -- | P4 | 14 | Ground (not used) | -- |
| | 5 | Ground (not used) | -- | | | | |
| | 6 | Ground (not used) | -- | | | Tied together | |
| | 7 | Ground | 4 | | | in cable | |
| | 8 | TTY In | 5 | | | +45V | 1 |
| | 9 | TTY Out | 7 | | | | 8 |
| P4 | 10 | Motor On Return | 11 | | | | 6 |

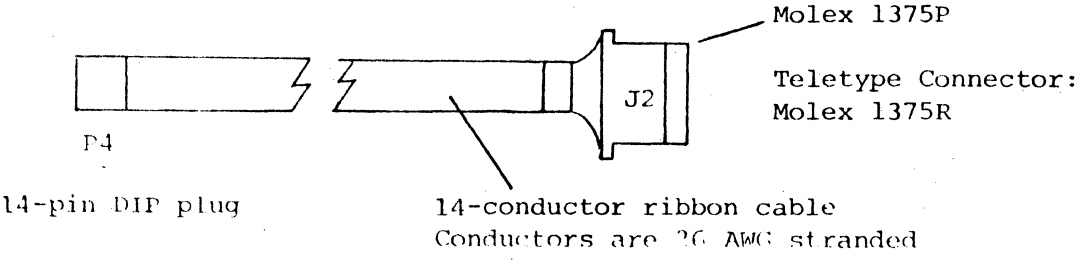



Figure 3-32 Teletype Cable Description

### 3.3.5.9 Programming Example

The Assembler Language statements shown in Table 3-5 demonstrate one method for using the Distributed I/O System. The technique is based on the programming information in Section 2.

The demonstration code falls into three distinct parts: Interrupt Locations, I/O Initiation and End-of-Block Service.

1. Interrupt Locations

   The first part of the code defines and fills the standard locations associated with IOP channel to which the device's PicoProcessor is connected. There are six locations involved of which two -- the Byte Count and the Buffer Location -- are dynamic for each I/O Initiation, while the reamining words can be fixed at program load time.

2. I/O Initiation

   The second part of the code accomplishes the transfer of one physical record each time the user's main-line program executes this instruction:

   ```
   CALL KEYIN
   ```

   It is assumed that the calling program has previously set up these two words:

   | | |
   |---|---|
   | BYTCNT | Number of bytes to be transferred |
   | BUFADD | Word address of record buffer |

   The demonstration code converts this information into the form required by an Automatic I/O instruction: negative byte count and byte address of the buffer minus one. Once these computations have been stored into the interrupt locations, the PicoProcessor is sent a command specifying:

   ```
   Begin at Branch Address +1
   ```

   Notice that the Device Address used in this sequence is coded with the same Device Address used in the interrupt location, plus one. The assembled object code will indicate the transfer of a command, rather than of data.

   No attempt is made in the demonstration code to overlap record transfer with CPU activity. Instead, a dead loop is issued -- JMP $ -- until the PicoProcessor interrupts not to the Data Service interrupt location, but to the End-of-Block location. At this point control passes to the subroutine labelled EOB.

3. End-of-Block Service

   The End-of-Block subroutine could perform error analysis, retry, etc. The demonstration code simply obtains device status from the PicoProcessor and passes status back to the main-line program through the A register.

Table 3-5. Programming Example

```
TYINT   EQU     :CW         STANDARD ITY/LRI INTERRUPT
TYDEVA  EQU     :FB         STANDARD ITY/LRI DEVICE ADDRESS
        ABS     TYINT
        AIB     TYDEVA
        DATA    $-$
        DATA    $-$
        DATA    W
*               INTERRUPT FOR END-OF-BLOCK
        JST     *$+1        CALL END-OF-BLOCK ROUTINE
        DATA    EOB
*
KEYIN   ENT     ENTRY POINT FOR KEYBOARD INPUT
        LDA     BYTCNT      # OF BYTES IN MESSAGE
        NAR                 AIB INSTRUCTION NEEDS NEGATIVE
        STA     TYINT+1     PUT IN AIB INSTRUCTION
        LDA     BUFADD      ADDRESS (WORD) OF BUFFER
        LLA     1           AIB INSTRUCTION NEEDS BYTE ADDRESS
        SAI     1           STARTS AT -1
        STA     TYINT+2     PUT IN AIB INSTRUCTION
        LDA     =:0618      WORD TO START PICOPROCESSOR
        OTA     TYDEVA+1    SEND TO PICO
        JMP     *           WAIT FOR END-OF-BLOCK
*
EOB     ENT     END-OF-BLOCK INTERRUPT SUBROUTINE
        INA     TYDEVA+1    INPUT STATUS
        RTN     KEYIN       RETURN TO CALLER WITH
*                           STATUS IN A REGISTER
```

### 3.3.6 CRT or Modem

#### 3.3.6.1 Description

Intelligent Cable 14630-01 provides an interface between an asynchronous EIA-type CRT, such as the ADDS Model 580 or equivalent, and an LSI Series computer.

Intelligent Cable 14630-02 provides an interface between an asynchronous Modem, such as the Bell Model 103 or equivalent, and the computer.

The Cable's PicoProcessor performs all data and control signal conversion (serial-to parallel and parallel-to-serial) required to control device operation and data transfer to and from the device.

The PicoProcessor controls device operation in half-duplex mode in which data is transmitted to or from the device, but not simultaneously. Use of the "Echo Incoming Data" feature causes input from the device to be automatically "echoed" back to the device for display without the need for separate software instructions to obtain a display.

The selection of an input or output operation and the selection of the Echo feature are made by setting the appropriate mode control bits of the Command Word. Additional bits of the Command Word are set to select character detection and parity bit standardization.

The Character Detection feature provides automatic detection of the 7-bit ASCII Carriage Return character (:0D). The PicoProcessor generates an End-of-Block interrupt when Carriage Return is received to terminate the input operation even though the input buffer (specified by the Auto I/O instruction) has not been filled.

When the Parity Bit Standardization feature is enabled, the most-significant bit of all incoming data characters (bit 7) is set to the same value independent of its actual value. In this way, all data characters can be examined in the same operation, whether they use odd or even parity.

Character length and parity format are selectable as described under 4.3.2.1. The baud rate is selectable in the IOD as described under 4.3.1.2. For additional information on device operation, refer to the device instruction manual and to the EIA RS232 specifications.

#### 3.3.6.2 Physical Details

Cable Lengths:
- PicoProcessor to IOD, 4 feet
- PicoProcessor to Device, 12 feet

Standard Channel Number
- CRT, 4 (Device Address field = :F8)
- Modem, 2 (Device Address field = :F4)

Standard Data Service Interrupt Address:
- CRT, :E0
- Modem, :D0

Standard End-of-Block Interrupt Address
- CRT, :E4
- Modem, :D4

Interface Line Description (see 3.3.6.6)

#### 3.3.6.3 Device Status Word

To control the transfer of data to and from the device, the PicoProcessor sequences through a series of operations based, in part, on the state of individual bits of the device Status Word. The Status Word, shown in figure 3-33, indicates certain operational and error conditions in the Device. When the PicoProcessor receives an input instruction requesting device status, it immediately transfers the entire Status Word to the CPU on bits 0 through

5 of the data bus. An EOB interrupt caused by an Overrun or Parity Error will not be indicated by the status word. If all status bits are in the proper state, the byte count at the appropriate memory location must be checked (for byte count = 0). If Character Detection is selected, the appropriate memory location must be checked for receipt of a Carriage Return character. If the byte count does not equal zero and if Character Detection is selected and no Carriage Return has been received, an Overrun or Parity Error exists.

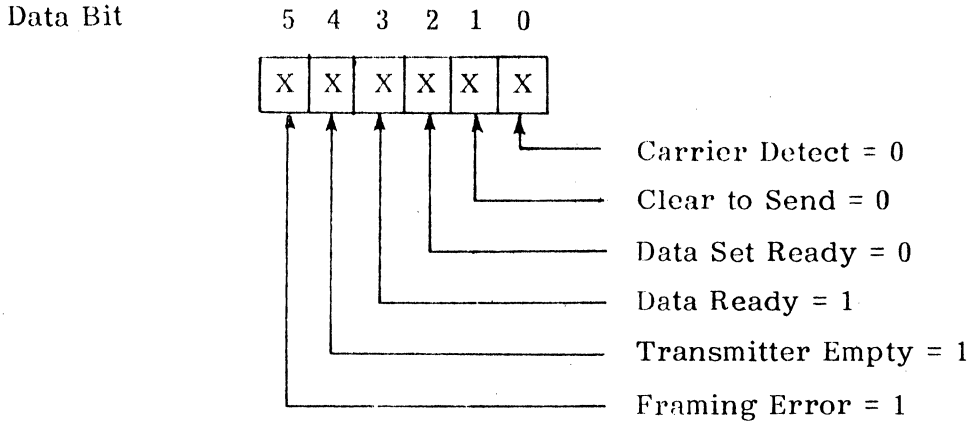



Figure 3-33. Device Status Word

Following is the definition of each status bit:

1. Carrier Detect. For the Modem Intelligent Cable, this line is driven true during an input operation by the Modem (Data Set) to indicate that it is receiving a signal which meets its suitability criteria. For the CRT Intelligent Cable this line is connected to the Data Terminal Ready output line of the CRT. The PicoProcessor monitors this line and the Data Ready line to determine when a full character has been received.

2. Clear to Send. For the Modem Intelligent Cable this line is driven true for an output operation by the Modem (Data Set) to indicate it is ready to transmit data. For the CRT Intelligent Cable, this line is connected to the Data Terminal Ready output of the CRT as is Carrier Detect. The PicoProcessor checks this line to determine if data can be transmitted.

3. Data Set Ready. This line is driven by a Modem only during an input or output. It indicates the status of the Modem. The true condition indicates that the Modem is connected to a communications channel and is not in test, talk or dial mode and that it has completed any applicable timing functions required to complete call establishment.

4. Data Ready. This line when true during an input operation indicates that an entire character has been received by the PicoProcessor Receiver. Data can now be transferred to the CPU. This line is reset after each data character has been transferred to the CPU.

5. Transmitter Empty. This line when true during an output operation indicates that the PicoProcessor Transmitter has serially transferred an entire character to the device. Another character can then be transferred from the CPU. This line is reset prior to each block transfer.

6. Framing Error. This status line when true indicates that a valid stop bit was not present in the received character during an input operation or that the Break Key has been pushed during an input or output. This line is reset prior to each block transferred and updated for each character input.

3.3.6.4 Mode Bit Significance

Certain PicoProcessor and device operations are selected by the mode bits (bits 0-3) of the Command Word. The bits are set at the beginning of an input or output operation. Once set, they cannot be changed until the operation is completed.

1. Bit 0 (Data Terminal Not Ready). This is the Data Terminal Not Ready line driven by the PicoProcessor indicating that the PicoProcessor is ready. This bit is reset to the Data Terminal Ready condition.

2. Bit 1 (Request to Send). This is the Request to Send bit. It is set to "1" to select a data output transfer. A "0" level selects an output transfer. The bit is used internally to control the PicoProcessor operating sequence to perform either an input or output operation. When reset this bit selects the input sequence.

3. Bit 2. This bit is not used by the CRT or Modem.

4. Bit 3 (Echo Incoming Data). A "1" in this bit selects the Echo mode of operation in which the serial input data is returned or "echoed" back to the device as it is received by the PicoProcessor. When reset, this bit selects the non-Echo mode.

3.3.6.5 Operating Sequence

The PicoProcessor has 16 unique sequence addresses (:0 through :F). When it receives a Command Word with the Begin bit set to "1", it immediately begins operation at the sequence address specified by the Branch Address field of the Command Word. Standard software enters the sequence at one of the following sequence addresses. All mode bits are "0" unless otherwise specified as a "1".

| Sequence Address | Operation |
|---|---|
| :1 | Start Input |
| :1 (Mode bit 1 = 1) | Start Output |
| :1 (Mode bit 3 = 1) | Start Input in Automatic Echo mode |
| :3 | Ring (Modem only). Wait for DSR (Data Set Ready); then generate EOB interrupt. Answer call and inform CPU. |
| :4 | Ring In (Modem only). Wait for DSR, then start input; generate EOB interrupt when complete. Answer call and start input sequence. |
| :4 (Mode Bit 1 = 1) | Ring Out (Modem only). Wait for DSR, then start output operation; generate EOB interrupt when completed. Answer call and start output sequence. |

3.3.6.5.1 Output Operation. Details of the CRT/Modem PicoProcessor operating sequence for an output operation are shown on the flow chart (figure 3-33). The operating sequence (firmware) controls the transfer of data from the CPU to the PicoProcessor. The actual transfer of data to the device is performed by a Serial Transmitter in the PicoProcessor which does the parallel-to-serial data conversion. Operation of the Transmitter does not appear on the flow chart, but the firmware times the generation of interrupts on the basis of signals from the Serial Transmitter.

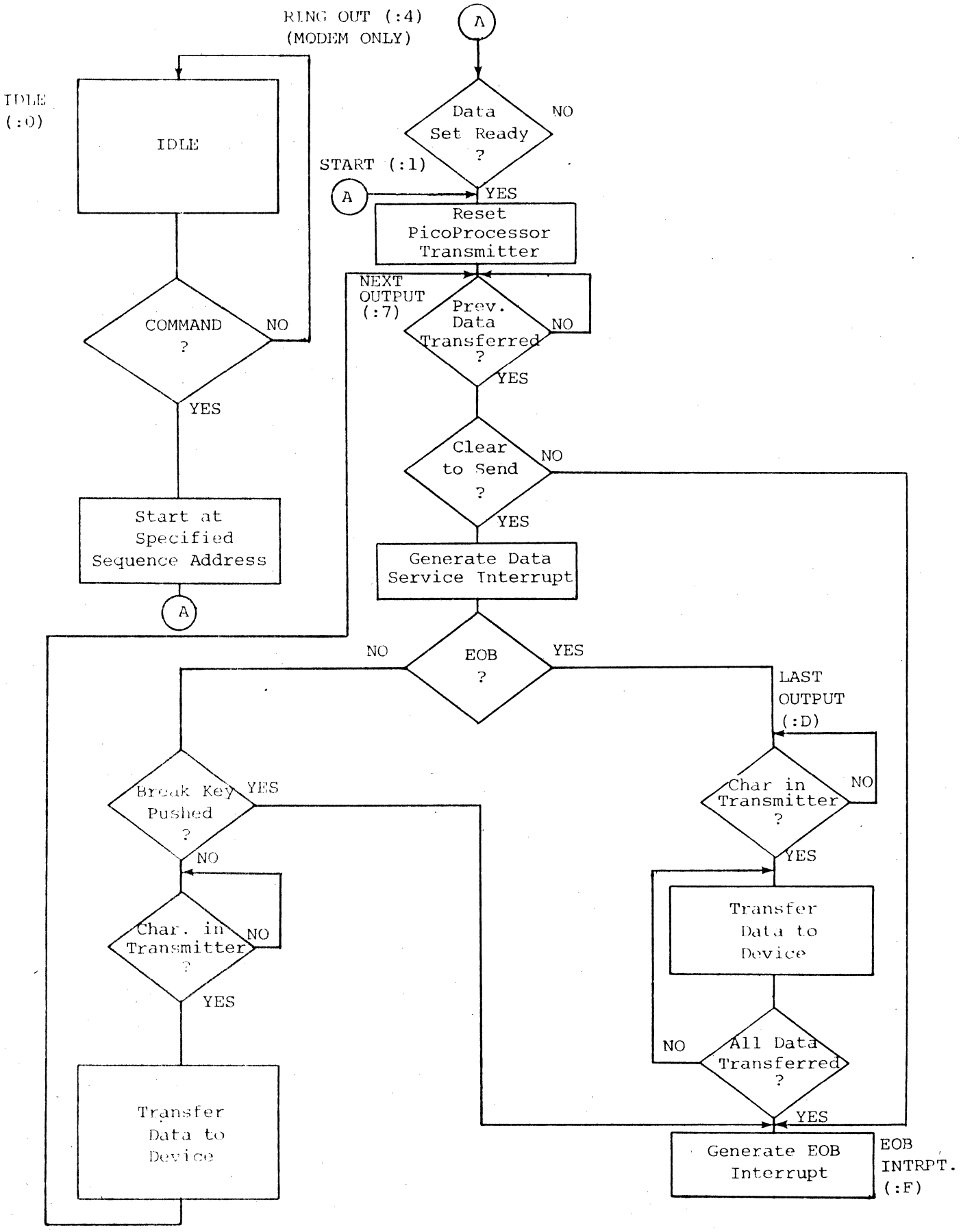



Figure 3-34. Firmware Sequence - CRT/Modem Output

Following is a description of the flow chart with each segment of the sequence identified by name and sequence address.

IDLE (:0)

The PicoProcessor is initially in the Idle state as a result of a Reset command or because of an End-of-Block interrupt. A Begin command starts PicoProcessor operation at the sequence address specified by the Branch Address field of the Command Word.

RING OUT (:4)

This starting sequence address is used by modems to start an output operation only after Data Set Ready (DSR) is received. The PicoProcessor waits for DSR to become true before proceeding with the output operation.

START (:1)

The output operation is started immediately when the sequence address of :1 is specified in the Branch Address field of the Command Word. The PicoProcessor generates a 500 ns Reset pulse to reset the PicoProcessor Transmitter.

NEXT OUTPUT (:7)

The PicoProcessor monitors the Transmitter Empty status line and waits for it to become true indicating that the transmitter is cleared of all data. It then monitors the Clear to Send (CTS) line from the device. If this line is true, indicating that the device is ready to accept data, the PicoProcessor generates a Data Service interrupt to transfer data from the CPU to the PicoProcessor Transmitter. If CTS is false, the PicoProcessor generates an End-of-Block interrupt.

If the data transferred (as a result of the Data Service interrupt) was not the last byte of data to be transferred, ,the PicoProcessor checks the Framing Error status line and generates an End-of-Block interrupt if the line is true (framing error exists). If the line is false, the PicoProcessor checks the Transmitter Empty line. The PicoProcessor waits for this line to go false indicating that the data transferred as a result of the Data Service interrupt is now in the transmitter. The PicoProcessor returns to the Next Output sequence and waits for Transmitter Empty to become true indicating that the entire character has been transferred to the device. The PicoProcessor then checks CTS and generates the next Data Service interrupt, as before.

LAST OUTPUT (:D)

When the last character has been transferred to the PicoProcessor, the PicoProcessor monitors the Transmitter Empty status line. It waits, first, for the line to become false indicating that the character from the CPU is now in the Transmitter. It then waits for the line to become true indicating that the entire data character has been output to the device.

EOB INTERRUPT (:F)

After Transmitter Empty becomes true at the end of the Last Output sequence or upon detection of a Framing Error or if Clear to Send is not received, the PicoProcessor generates an End-of-Block interrupt. The PicoProcessor returns to the Idle state service of the interrupt.

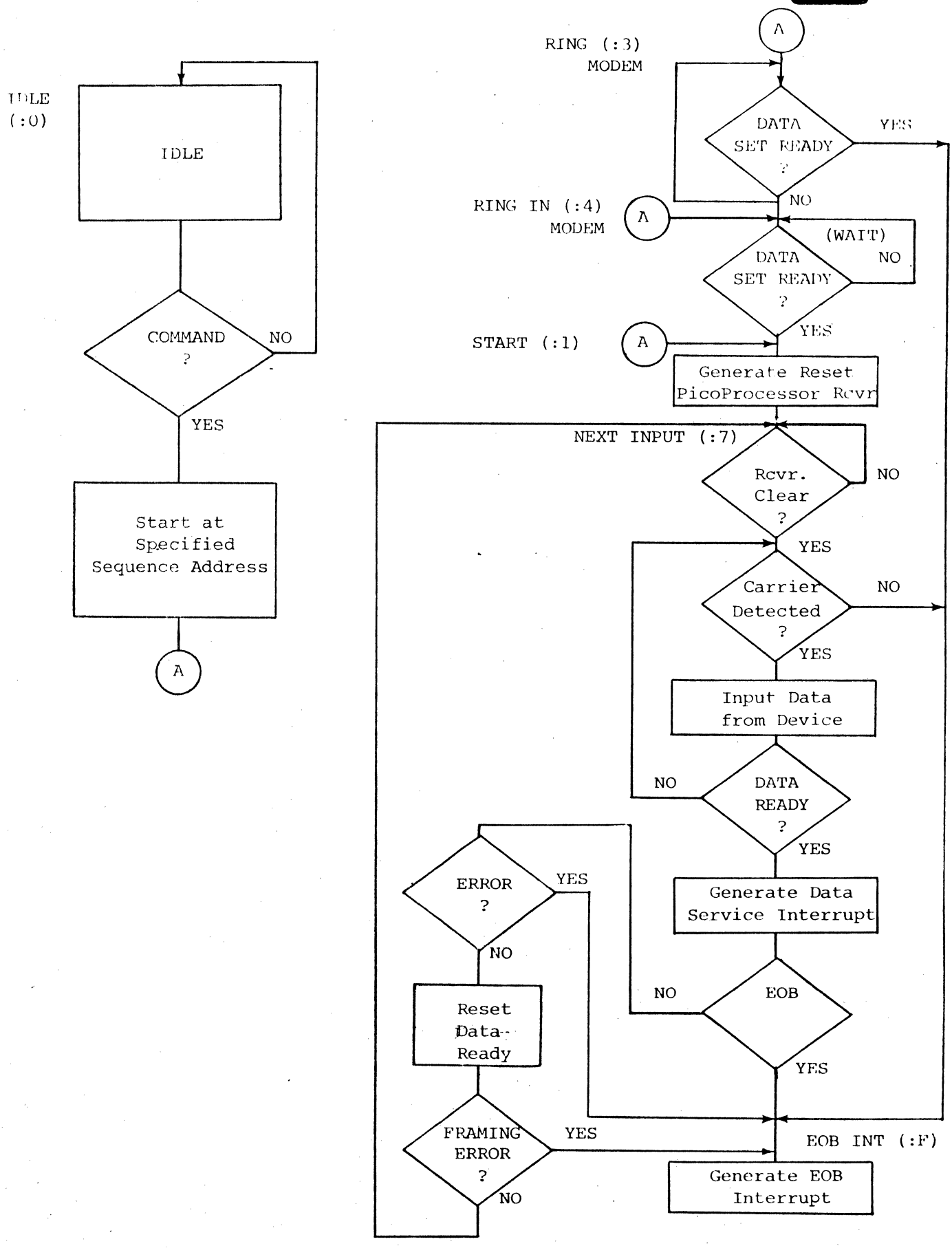



Figure 3-35. Firmware Sequence - CRT/Modem Input

3.3.6.5.2 Input Operation. Details of the CRT/Modem PicoProcessor operating sequence for an input operation are shown in a flow chart (figure 3-34). The firmware sequence controls the transfer of data from the PicoProcessor to the CPU. The actual transfer of data from the device to the PicoProcessor is controlled by a Receiver in the PicoProcessor which does the serial-to-parallel data conversion. Operation of the Receiver does not appear in the flow chart but firmware times the generation of interrupts on the basis of signals from the Receiver.

Following is a description of the flow chart with each segment of the sequence identified by name and sequence address.

IDLE (:0)

The PicoProcessor is initially in the Idle state as a result of a Reset command or because of the completion of an End-of-Block interrupt. A Begin command starts operation at the sequence address specified by the Branch Address field of the Command Word.

RING (:3)

This starting sequence address is used only by modems for a "ring and answer" operation. A ring is generated to the modem and it answers by returning DSR (Data Set Ready). Upon receipt of DSR, the PicoProcessor generates an End-of-Block interrupt. Further modem operations are then dependent on instructions from the CPU.

RING IN (:4)

This starting sequence address is used only by modems. In this case, a ring is generated to the modem and, when answered by DSR, the PicoProcessor immediately starts an input operation.

START (:1)

The input operation starts at this point. The PicoProcessor generates a 500 ns Reset pulse to reset the PicoProcessor's Receiver.

NEXT INPUT (:7)

The PicoProcessor monitors the Data Ready line and waits for the line to become false to indicate that the PicoProcessor Receiver is cleared of all data. It then checks the Carrier Detect status line. This line is true while input serial data is being received. If false, the PicoProcessor generates an End-of-Block interrupt. If the line is true, the PicoProcessor next monitors the Data Ready line which becomes true when an entire input character has been input to the PicoProcessor Receiver. If the Data Ready line is false (serial data is still being received), the PicoProcessor rechecks Carrier Detect and generates an End-of-Block interrupt if it is false -- since the carrier signal is no longer present.

When the Data Ready line becomes true, the PicoProcessor generates a Data Service interrupt to transfer the received character to the CPU. If the character is not the last to be transferred, the PicoProcessor checks the character for Parity or Overrun Error, resets Data Ready and then checks for a Framing Error. In each case, the PicoProcessor generates an End-of-Block interrupt on a positive response. If no errors exist, the Pico-Processor returns to be Next Input sequence where it inputs the next character. This continues until the byte count = 0.

EOB INTERRUPT (:F)

If the byte count = 0 after a Data Service interrupt, it indicates that all data has been input and the PicoProcessor generates an End of-Block interrupt. The PicoProcessor also generates an End-of-Block interrupt when an Overrun or Parity Error or Framing Error is detected or if Carrier Detect status is false. When the interrupt is serviced, the PicoProcessor returns to the Idle state.

3.3.6.6 Interface Description

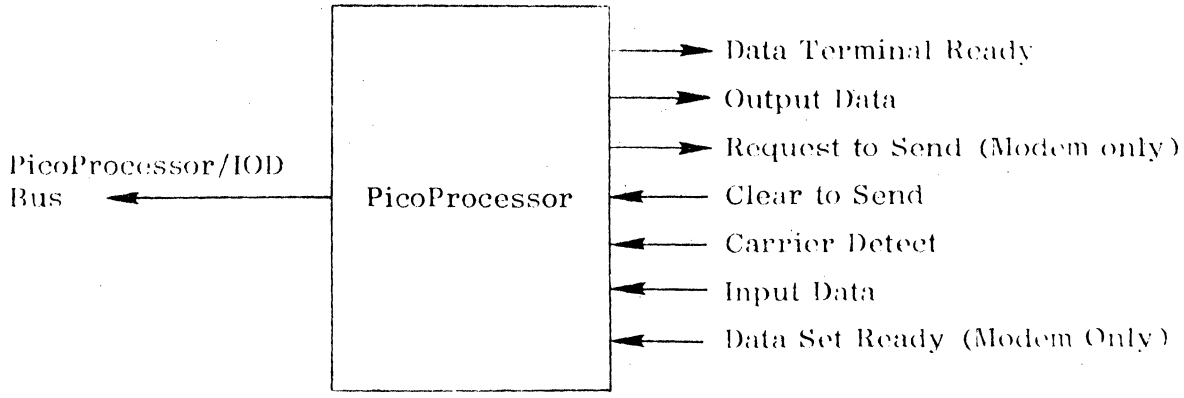


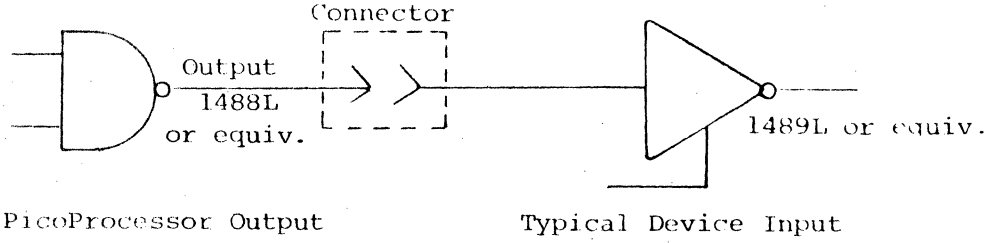

Figure 3-36. PicoProcessor Interface

Interface lines between the PicoProcessor and the CRT or Modem (figure 3-36) include one serial data output line, one serial data input line, two control lines to the device, and three status lines from the Modem and two from the CRT. Interface timing for the CRT is shown in figure 3-37. Interface timing for the Modem is shown in figure 3-38 (also see figure 3-29 for data format).

The output lines are standard RS232 interface drivers. Logic levels are as follows:

Data Output: +12V = Space = 0
-12V = Mark = 1

Control Output: +12V = assertion = 1
-12V = non-assertion = 0

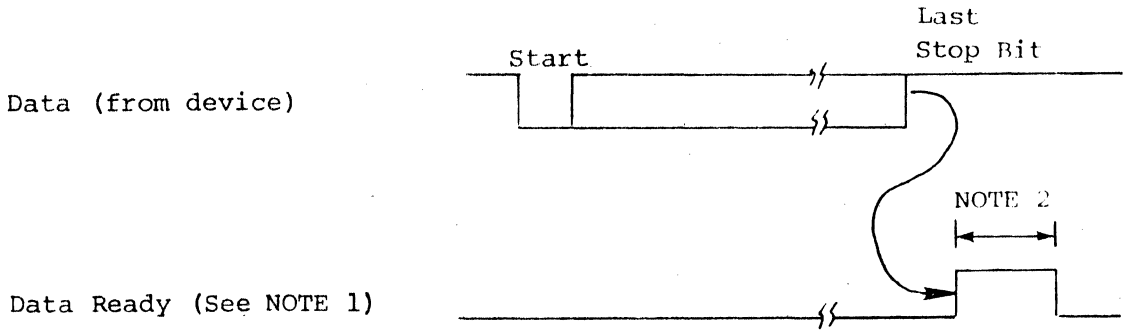



Input lines are standard RS232 interface receivers. Logic levels are as follows:

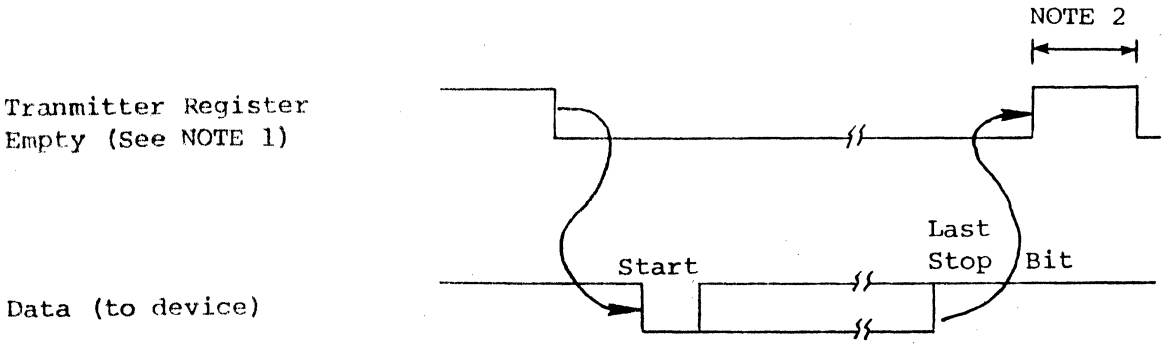
Data Input: -3V to -25V = Mark = 1
+3V to +25V = Space = 0

Control Input: +3V to +25V = assertion = 1
-3V to -25V = non-assertion = 0

INPUT FROM DEVICE

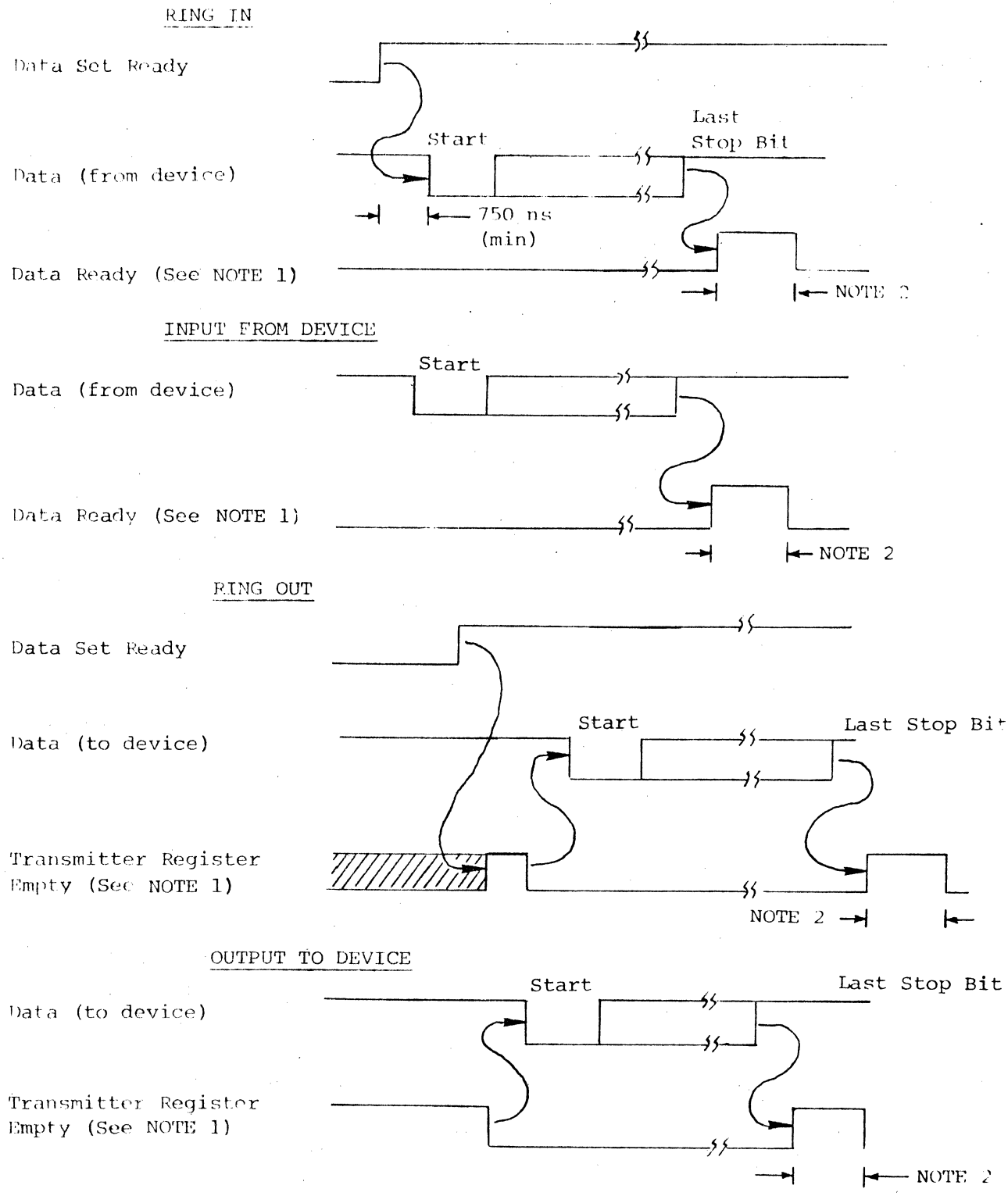



OUTPUT TO DEVICE



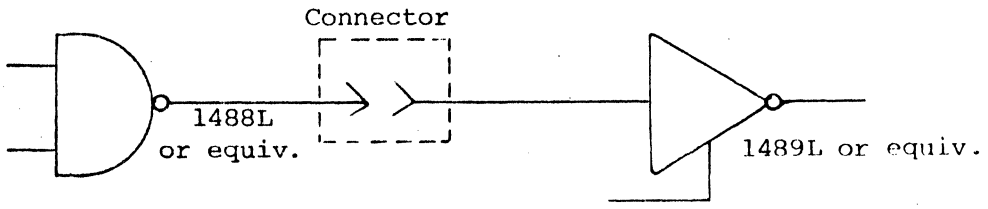

NOTE 1: This line is a status line available only to the Programmer.
NOTE 2: Minimum time equal to the instruction time of the Auto I/O byte instruction plus 4 μsec. For instruction time, see the appropriate Computer Handbook.

Figure 3-37. Interface Timing - CRT Intelligent Cable




NOTE 1: This is a status line available only to the Programmer.
NOTE 2: Minimum time equal to the instruction time of the Auto I/O byte instruction plus 4 μsec. For instruction time, see the Appendix of the appropriate Computer Handbook.

Figure 3-38. Interface Timing - Modem Intelligent Cable




Interface lines are described below:

1. DTR (Data Terminal Ready). This line is driven positive-true by the PicoProcessor to indicate that the PicoProcessor is ready to accept or send device data. This line is reset to the true state indicating the Data Terminal is ready. See section 3.3.6.4. Modes, for further definition.

2. XMIT (Output Data). This line carries serial output data to the device from the PicoProcessor's Transmitter.

3. RTS (Request to Send). This line is driven positive-true to the Modem by the PicoProcessor to request an output operation. This lines is not used by the CRT. See section 3.3.6.4, Modes, for further definition.

4. CTS (Clear to Send). This line is driven positive-true by the device to indicate that the device is ready to accept data. See Device Status Word, section 3.3.6.3. for further definition.

5. CDET (Carrier Detect). This line is driven positive-true by the device to indicate that it is receiving a suitable signal. See Device Status Word, section 3.3.6.3. for further definition.

6. REC (Input Data). This line carries serial input data from the device to the PicoProcessor's Receiver.

7. DSR (Data Set Ready). This line is driven positive-true by a Modem and indicates the status of the Modem. See Device Status Word, section 3.3.6.3, for further definition.

3.3.6.7 Strapping Requirements - Same as teletype, see figure 3-31.

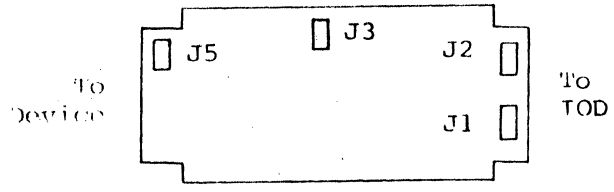



Figure 3-39. Connector Locations - CRT/Modem PicoProcessor

### 3.3.6.8 Device Cable Description

The device cable is 12 feet long and terminated at the PicoProcessor end with one 14 pin DIP plug (P5). It is terminated on the device end by a 25 pin connector that mates with connector J1 on the device. Figure 3-40 lists all interface lines in the cable and identifies the connectors used. The location of the mating connector on the PicoProcessor is shown in figure 3-39.

| PicoProcessor | | Description | CRT | | Modem | |
|---|---|---|---|---|---|---|
| Conn | Pin | | Conn | Pin | Conn | Pin |
| P5 | 1 | Clear to Send | J1 | 20 | J1 | 5 |
| | 2 | Data Terminal Ready | | 5 & 8 | | 20 |
| | 3 | Data Set Ready (Modem only) | | -- | | 6 |
| | 4 | Transmitted Data | | 3 | | 2 |
| | 5 | Request to Send (Modem only) | | -- | | 4 |
| | 6 | Carrier Detect | | 20 | | 8 |
| | 7 | Received Data | | 2 | | 3 |
| | 8 | Ground | | 1 & 7 | | 7 |
| | 9 | Ground (not used) | | -- | | -- |
| | 10 | Ground (not used) | | -- | | -- |
| | 11 | Ground (not used) | | -- | | -- |
| | 12 | Ground (not used) | | -- | | -- |
| | 13 | Ground (not used) | | -- | | -- |
| P5 | 14 | Ground (not used) | J1 | -- | J1 | -- |

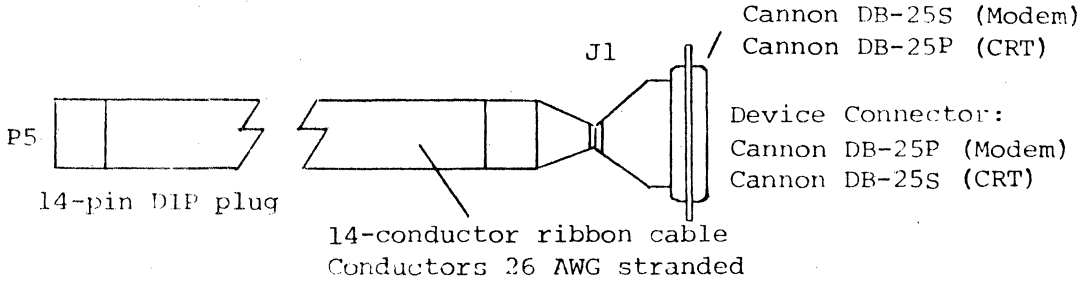



Figure 3-40. CRT/Modem Cable Description

### 3.3.6.9 Programming Example

For an example of programming the CRT/Modem, see the Programming Example for the Teletype (Table 3-5). Only the device address and interrupt address will be different. These addresses are listed under 3.3.6.2.

### 3.3.7 General-Purpose Intelligent Cables

#### 3.3.7.1 General Description

The General-Purpose Cables 14631-11 (negative true interface logic) and 14631-12 (positive true interface logic) allow a user to interface most low- to medium-speed peripheral or special purpose input/output devices to an LSI Series Computer without the need to design and implement special interface logic. Both Intelligent Cables operate in a similar manner and the same software is used to drive either cable. All modification of signal polarity is handled by the device's PicoProcessor. The General-Purpose Intelligent Cable is especially designed to operate devices having simple two-wire "handshake" or strobed I/O discipline--but its expandable interface capability also provides for the control and operation of peripheral devices having more complex I/O disciplines.

#### 3.3.7.2 Specifications

**Data Types:** Eight-bit parallel output data with storage; Eight or 16-bit parallel input data.

**Operating Modes:** Simplex and half-duplex

**I/O Disciplines:** Two types:
1. Simple two-wire "handshake".
2. Strobe Data/Pulse Acknowledge; Start Pulse/Data Acknowledge.

Additional lines provided for data mode selection, device status, reset and end-of-transfer indicator.

**Data Modes:** Four modes selectable by software.
1. Data output (8 bits)
2. Data input (8 or 16 bits)
3. Command output (8 bits)
4. Extended status input (8 or 16 bits)

**Interface Logic:** Two types, positive true and negative true:

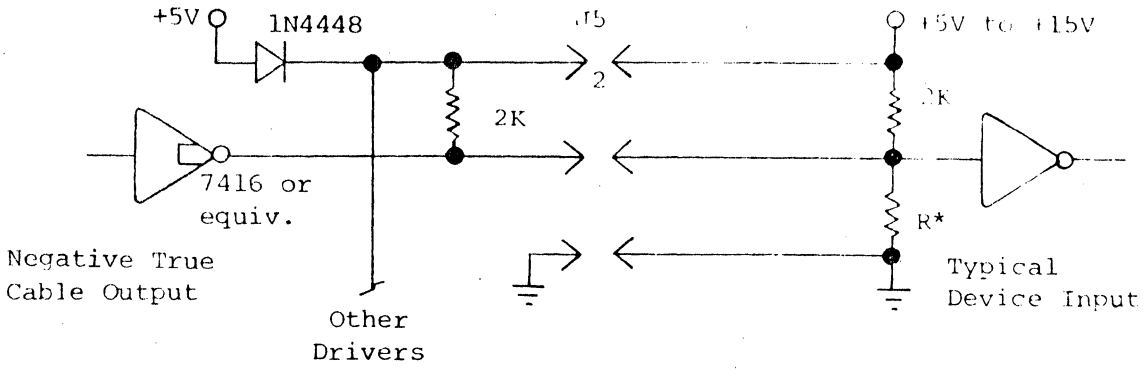
1. Intelligent Cable 14631-11, negative true:

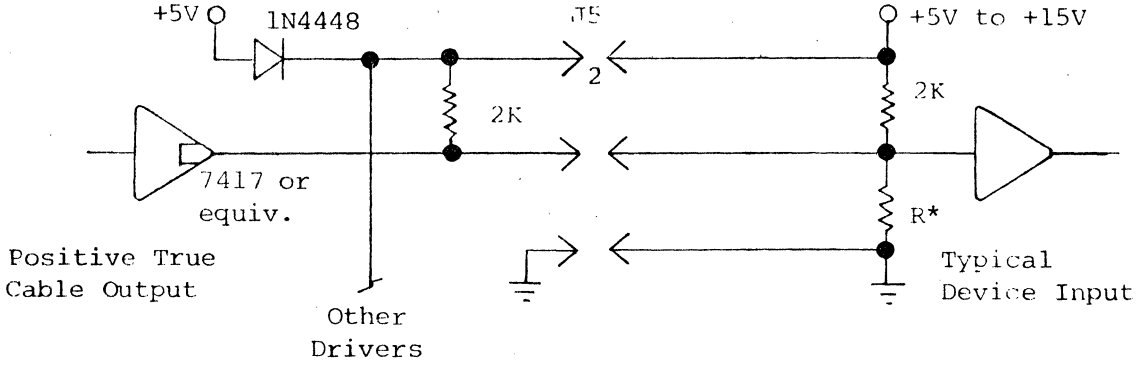
| Output lines | Input lines |
|---|---|
| Logical 0 = +5 to +15 volts | Logical 0 = +5 volts |
| Logical 1 = 0 volts | Logical 1 = 0 volts |

2. Intelligent Cable 14631-12, positive true:

| Output lines | Input lines |
|---|---|
| Logical 0 = 0 volts | Logical 0 = 0 volts |
| Logical 1 = +5 to +15 volts | Logical 1 = +5 volts |

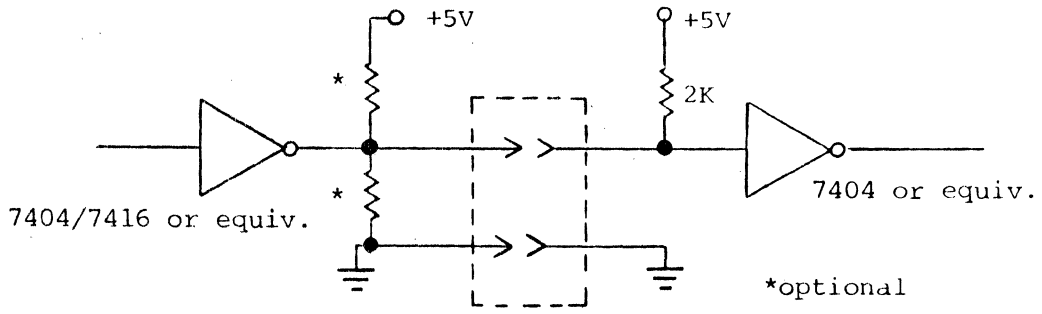
**Output lines:** Open-collector TTL drivers capable of sinking up to 32 ma at 15 volts.




*Optional; recommended for noise immunity if high current required.

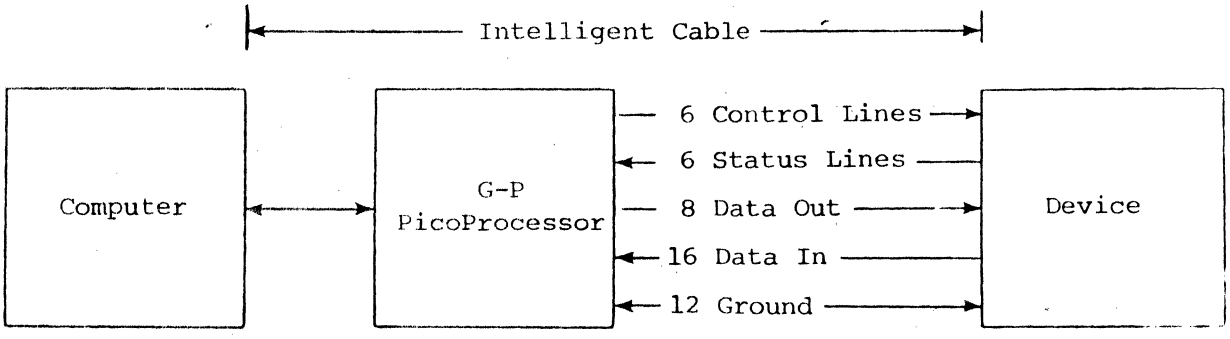
Input lines: One TTL load plus a 2K pullup resistor to +5V.




Typical Device Output G-P Intelligent Cable Input

PicoProcessor Microcycle Time: 250 ns

Interface Lines:
Output control (six lines)
Input status (six lines, expandable); see 3.3.7.3 (4)
Output data (eight lines)
Input data (16 lines)
Ground (11 Lines)
External Voltage (1 Line)



Physical Details: Cable Lengths:
IOD to PicoProcessor, 10-1/2 feet
PicoProcessor to Device, 1-1/2 feet
(no device connector supplied)

Standard Device Address/Interrupt Addresses:
(None assigned)
Dependent on IOD channel used

3.3.7.3 Software Considerations

As explained in Section 2, all operations are begun by outputting a Command Word ing to the selected device's PICOPROCESSOR. This is done by executing an output instructi with the control bit (bit 0) set to "1". The format of the Command Word is repeated here (Figure 3-41) in order to clarify operations unique to the G-P Intelligent Cable.

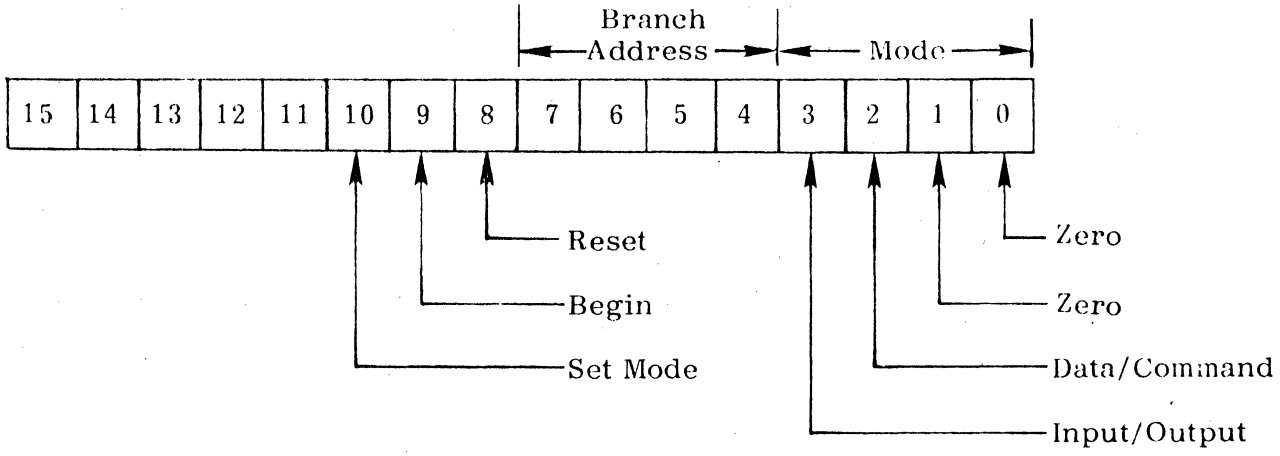



Figure 3-41. Command Word - G-P Intelligent Cable

3.3.7.3.1 Mode Field/Set Mode. The four operating modes are selected by the combination of states of mode bits 2 and 3. In order for these bits to be significant, Set Mode, bit 10, must be set to "1". The mode is then selected as follows:

| Input/Output Bit 3 | Data/Command Bit 2 | Mode |
|---|---|---|
| 0 | 0 | Output Command |
| 0 | 1 | Output Data |
| 1 | 0 | Input Status (extended) |
| 1 | 1 | Input Data |

Operation in each of the modes can be accomplished with Auto I/O instructions or by use of conventional Programmed I/O. Following are descriptions of each of the modes.

1. Data/Output (Mode bit 2 = 1, bit 3 = 0). In the Data Output mode, the G-P PicoProcessor can transfer eight-bit bytes of parallel data to the peripheral device. Data polarity is automatically inverted for negative true interface logic and left unchanged for positive true interface logic.

2. Data Input (Mode bit 2=1, bit 3=1). In the Data Input mode, the G-P PicoProcessor can input 8-bit or 16-bit parallel data from the peripheral device.

3. Command Output (Mode bit 2=0, bit 3=0). For operation with peripheral devices requiring additional commands other than normal handshake/strobe controls, the General Purpose PicoProcessor can send eight bits of command information to the device. Decoded, this provides up to 256 separate commands. If more than one device is connected to a single PicoProcessor, the eight bits of command information can be divided into two basic fields. For example, four bits could be decoded into 16 individual commands. The commands are sent over the data output lines and distinguished from data by the state of the Command/Data control line.

4. Extended Status Input (Mode bit 2=0, bit 3=1). The G-P PicoProcessor can input six lines of device status or, using its extended status capability, it can input up to 8 or 16 bits of status on the input data lines. This extended capability can be used by complex peripherals requiring additional status beyond that normally provided in the six-bit basic status word or when more than one device or function is connected to a single Intelligent Cable. Extended status is received on the input data lines with and distinguished from data by the state of the Command/Data control line.

3.3.7.3.2 Branch Address Field/Begin. The Branch Address field selects the starting address of particular operations in the PicoProcessor's firmware sequence. This field is significant only when the Begin bit (bit 9) of the Command Word is set to "1". In Auto I/O operations the entire firmware sequence is usually performed. In this case, status lines are checked, control lines to the device are asserted and interrupts generated for data service and for end-of-block. All of these functions are performed automatically. The user may use or not use particular control lines and can supply device status only as required by his particular device. Refer to Table 3-6(a) and (b) for the grounding requirements of any unused lines. The entire firmware sequence is described in 3.3.7.8.

When using Programmed I/O, the starting firmware address in the Branch Address field is generally used to access specific operations within the sequence. This allows the Programmer to assert specific device control lines. Generally, device status is first checked by software and the CPU then controls the timing of the assertion of the appropriate control lines. Programmed I/O Operation is described in more detail in 3.3.7.8.3.

3.3.7.3.3 Reset Bit. When this bit (bit 8) is set to "1", the device's PicoProcessor is reset to the Idle state, a Reset line to the device is asserted and the mode bits are reset to zero (Output Command mode).

NOTE: The reset line to the device is a negative true signal for both the 14631-11 and 14631-12 Intelligent Cables.

3.3.7.3.4 Device Status Word. The device's PicoProcessor sequences through a series of operations based, in part, on the state of individual bits of the device Status Word. The Status Word, shown in figure 3-42, indicates certain operational or error conditions within the device. When the PicoProcessor receives a status input instruction, it transfers the entire Status Word to the CPU. The status bits are described in detail in 3.3.7.7.2. Status polarity is left unchanged between the CPU and the device for negative true interface logic and automatically inverted for positive true interface logic.

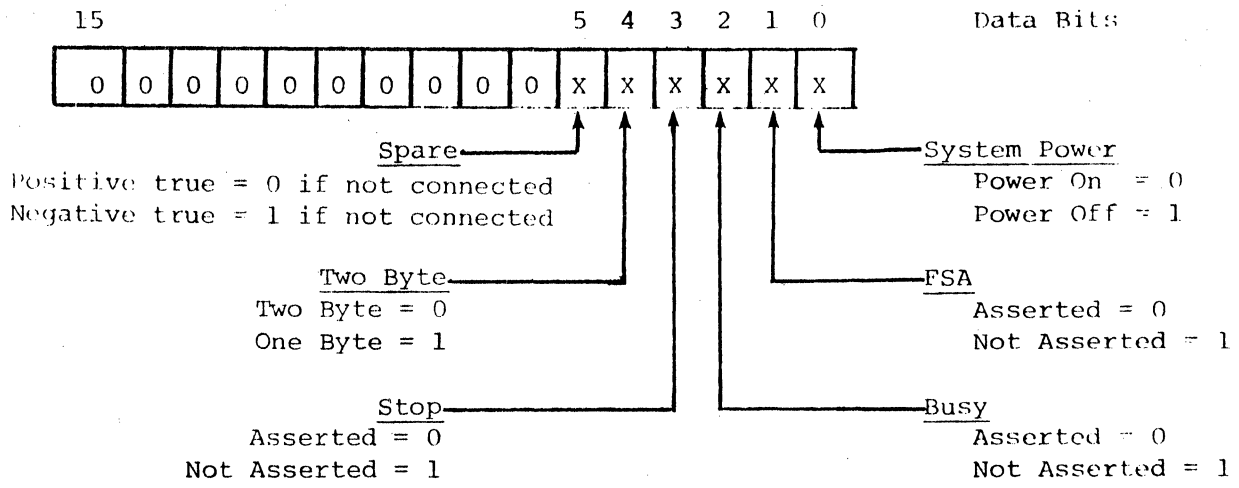



Figure 3-42. Device Status Word (CPU)

The status bits are described briefly below. They are described in greater detail in 3.3.7.7.2.

Function Strobe Acknowledge (FSA). The device drives this line (logical one) to indicate that it has sampled output data or has input data ready for transfer.

Two-Byte. This line indicates to the PicoProcessor that the device is either a one-byte device (logical zero) or a two-byte device (logical one).

Stop. The device drives this line (logical one) to indicate that, because of device end, no data transfer should occur.

Busy. The device drives this line (logical one) to indicate that it is not ready to receive or transfer data.

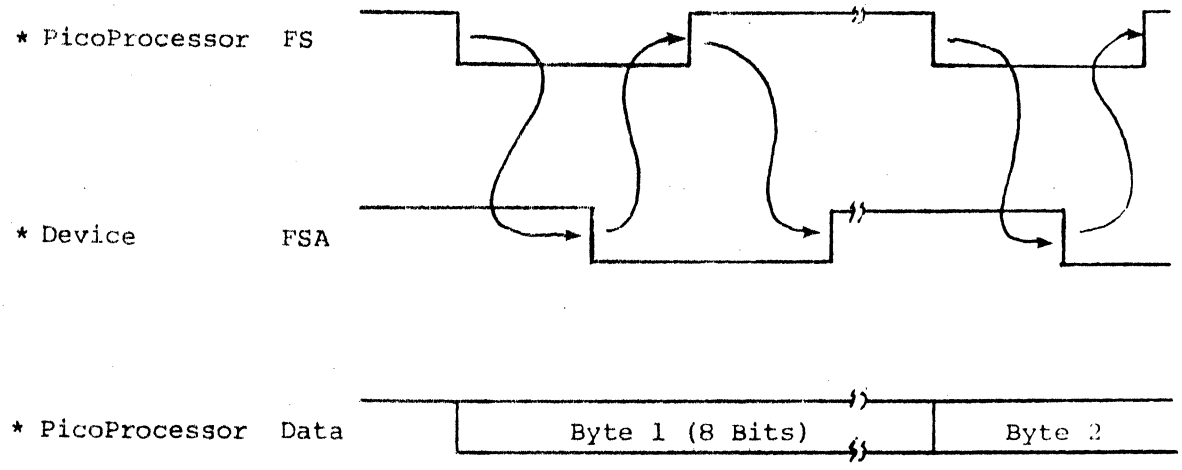
System Power. This line is driven (logical one) by the device to indicate that device power is turned on, the device cable is connected and the device is operational.

Spare. This line can be used by the device to indicate any desired status. If not used, this line provides a logical zero with positive true interface logic and a logical 1 with the negative true interface logic. Refer to Tables 3-6(a) and (b).

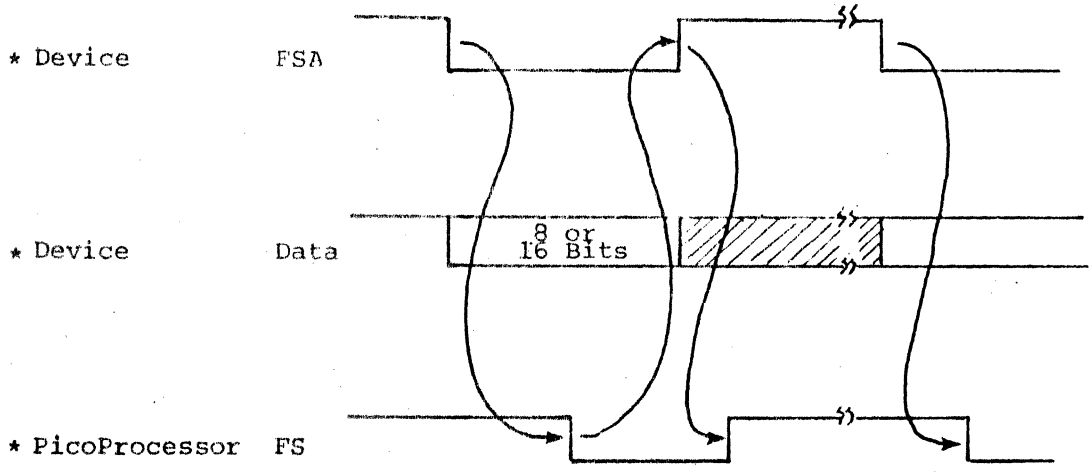
### 3.3.7.4 Use of Interface Lines

As explained in the previous paragraphs, the General-Purpose PicoProcessor's firmware sequence consists of a series of status checks and the automatic sequencing of control lines to the device. In order for the PicoProcessor's firmware to function properly, all unused status lines must be connected in accordance with Table 3-6(a) for negative true interface logic or Table 3-6(b) for positive true interface logic.

Input/Output transfers to or from a device having a simple two-line "handshaking" device I/O discipline require only the use of the Function Strobe control line (FS) and the Function Strobe Acknowledge (FSA) status line from the device. (Several



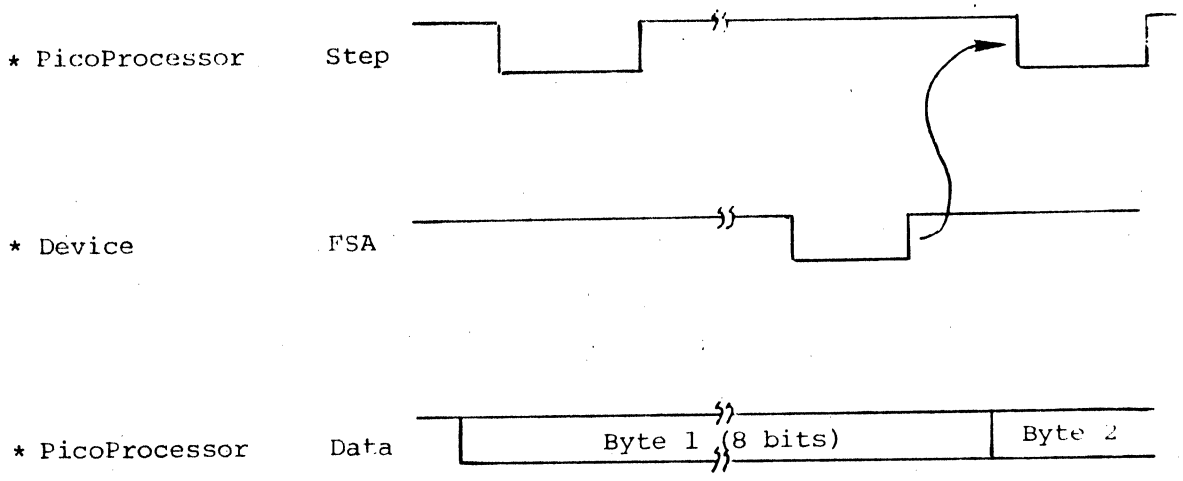

(a) Output



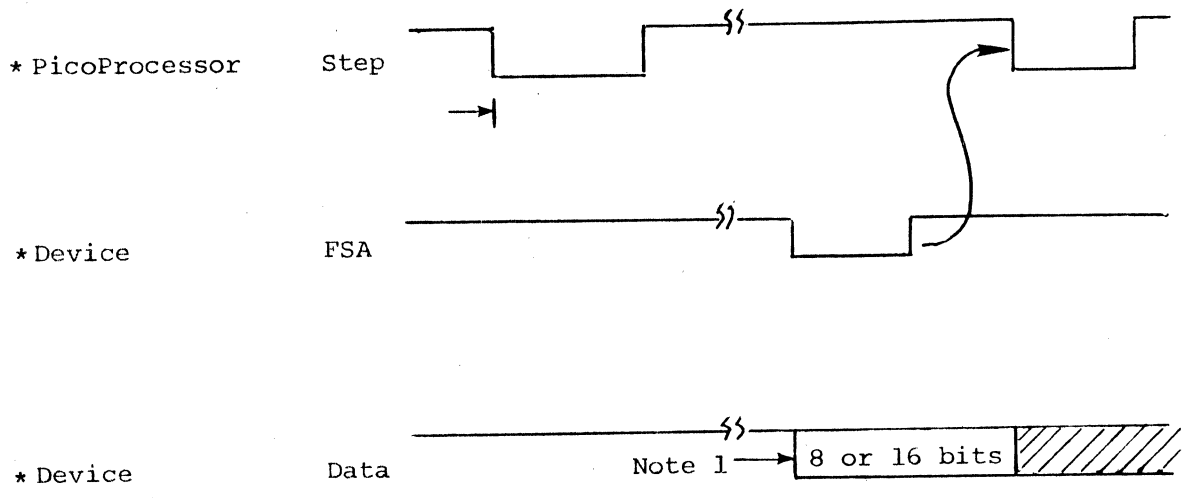

(b) Input

NOTE 1: Input/Output driven by software.
NOTE 2: For timing details, see figures 3-54(a) and 3-56(a).
*: Inverted For Positive True Interface Logic

Figure 3-43. I/O Transfer - Simple Two-Wire I/O Handshake Device Interface (Negative True Shown)




(a) Output




(b) Input

NOTE 1: Stable for Auto I/O instruction time +4 μs See appropriate CPU Manual for instruction times.

NOTE 2: For timing details, see figures 3-54(b) and 3-56(b).

*: Inverted For Positive True Interface Logic

Figure 3-44. I/O Transfer - Simple Two-Wire I/O Pulse-type Device Interface (Negative True Shown)

lines are mentioned briefly here and described in detail in 3.3.7.7.) This type of data transfer is shown in the timing diagram (Figure 3-43). For an output operation, Function Strobe is asserted until FSA is returned by the device to indicate that the output data lines have been sampled. For an input operation, FSA is driven by the device to indicate that the device has placed data on the lines. Function Strobe is driven by the Intelligent Cable to signal that data has been sampled from the input lines.

Input/output data transfers to or from devices having two-line pulse-type I/O discipline (non-handshake) also require the use of only two lines. The Step line is asserted to produce a 500 ns pulse to strobe data into the device in an output operation. In an input operation, the Step line signals the device to place data onto the data lines. In each case, the device responds with an FSA pulse to acknowledge that data has been accepted by the device during an output operation; or during an input operation, to indicate that data may be sampled. This type of transfer is shown in the timing diagram (Figure 3-44).

As a summary of interface line usage, Tables 3-6(a) and 3-6(b) list all control and status lines and describe their use with devices having either "handshake" or pulse-type I/O disciplines. In the table, "opt" indicates that use of the line is optional. That is, it can be used to improve performance or capability as required, but is not necessary. "Yes" indicates that use of the line is required for that operation using the indicated device. The connection notes for unused lines must be observed to assure proper sequencing of the PicoProcessor firmware.

Table 3-6(a)

Negative True Interface Line Usage

| | I/O Discipline/Operation | | | |
|---|---|---|---|---|
| | Handshake | | Pulse | |
| Status: | Output | Input | Output | Input |
| FSA | yes | yes | yes | yes |
| Stop | opt | opt | opt | opt |
| Busy | opt | opt | opt | opt |
| Power | opt (1) | opt | opt (1) | opt |
| Two-Byte | not used | opt (2) | not used | opt (2) |
| Spare | opt | opt | opt | opt |
| Control: | | | | |
| FS | yes | yes | opt | opt |
| Step | opt | opt | yes | yes |
| End | opt | opt | opt | opt |
| Input/Output | opt | opt | opt | opt |
| Data/Command | opt | opt | opt | opt |
| Reset | opt | opt | opt | opt |

NOTES: 1. Must be grounded if not used.
2. Ground for two-byte (16-bit) device.

Table 3-6(b)

Positive True Interface Line Usage

| | I/O Discipline/Operation | | | |
|---|---|---|---|---|
| | Handshake | | Pulse | |
| Status: | Output | Input | Output | Input |
| FSA | yes | yes | yes | yes |
| Stop | opt (1) | opt (1) | opt (1) | opt (1) |
| Busy | opt (1) | opt (1) | opt (1) | opt (1) |
| Power | opt (2) | opt | opt (2) | opt |
| Two-Byte | not used | opt (3) | not used | opt (3) |
| Spare | opt | opt | opt | opt |
| Control: | | | | |
| FS | yes | yes | opt | opt |
| Step | opt | opt | yes | yes |
| End | opt | opt | opt | opt |
| Input/Output | opt | opt | opt | opt |
| Data/Command | opt | opt | opt | opt |
| Reset | opt | opt | opt | opt |

NOTES:
1. Must be grounded if not used.
2. Must be open if not used.
3. Open for two-byte (16-bit) device.

3.3.7.5 Configurations

3.3.7.5.1 Single Intelligent Cable. Figures 3-45, 46 and 47 show typical configurations using a single Intelligent Cable with single simplex and half-duplex devices and with multiple simplex and half-duplex devices.

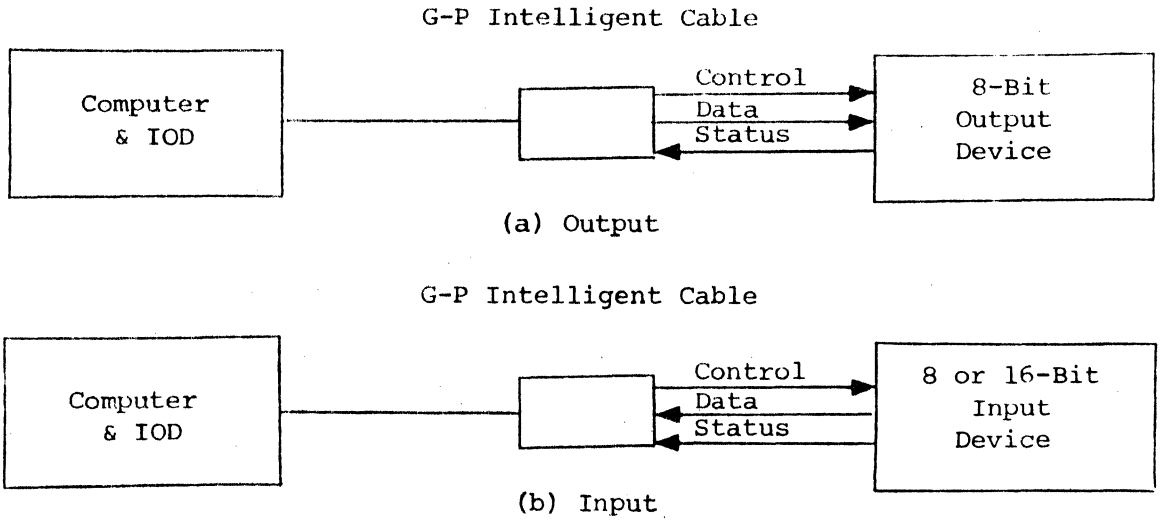



Figure 3-45. Simplex Devices

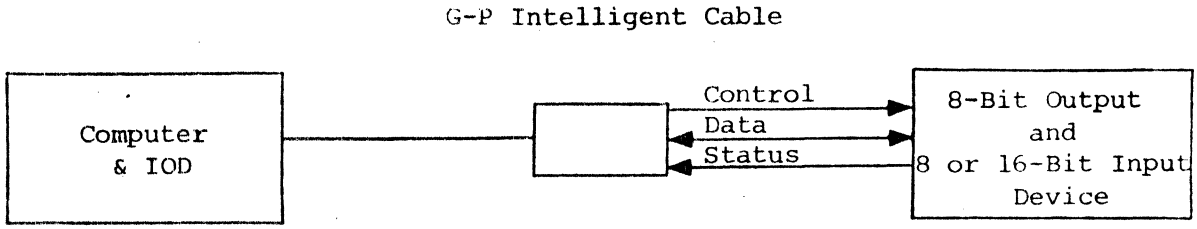



Figure 3-46. Half-Duplex Device

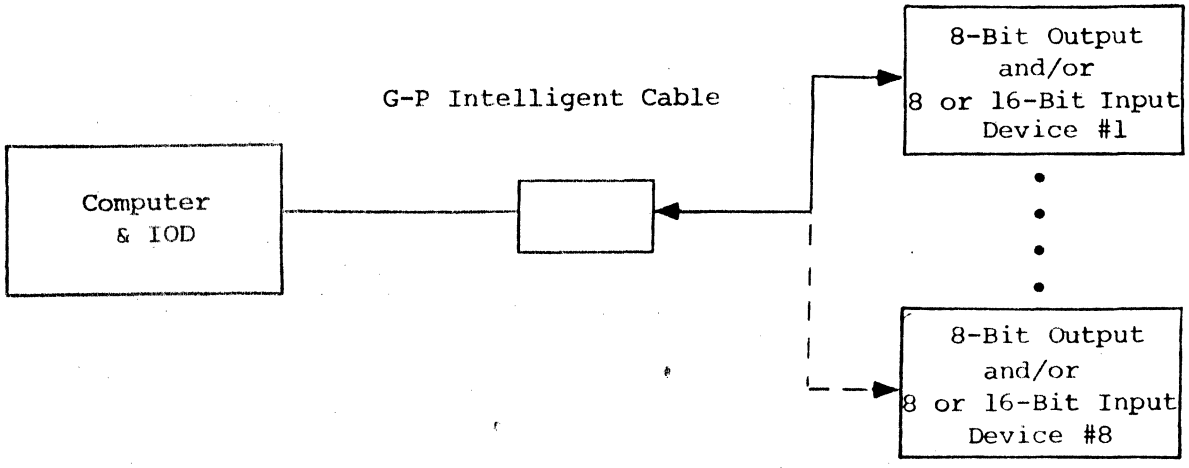



Figure 3-47. Multi-Simplex/Half-Duplex Devices

3.3.7.5.2 Multiple Intelligent Cables. The General-Purpose Intelligent Cables can also be operated in a multiple configuration. Multiple Intelligent Cables can be connected together to operate as a single Intelligent Cable with a wider parallel device data path (figure 3-48). The data path can be increased in increments of eight bits of output data and increments of eight or sixteen bits of input data. The data expansion capability is listed below. Multiple Intelligent Cable operation is not supported by Auto I/O instructions.

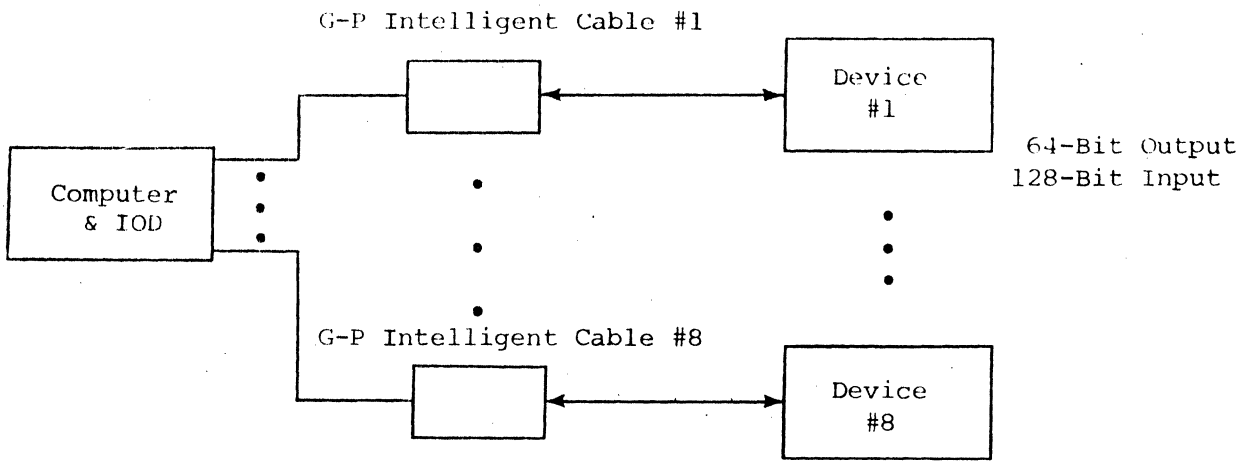



Figure 3-48. Multiple Intelligent Cables

| Number of Intelligent Cables | Input Bits | Output Bits |
|---|---|---|
| 1 | 8 or 16 | 8 |
| 2 | 24 or 32 | 16 |
| 3 | 40 or 48 | 24 |
| 4 | 56 or 64 | 32 |
| 5 | 72 or 80 | 40 |
| 6 | 88 or 96 | 48 |
| 7 | 104 or 112 | 56 |
| 8 | 120 or 128 | 64 |

3.3.7.6 Typical Applications

Figures 3-49 through 3-52 illustrate some of the applications in which use of the G-P PicoProcessor can be advantageous.

3.3.7.7 Device Interface Line Descriptions

Interface lines between the G-P Intelligent Cable and the device include six control lines to the device, six status lines from the device, eight output data lines and 16 input data lines. Following are descriptions of each of the lines.

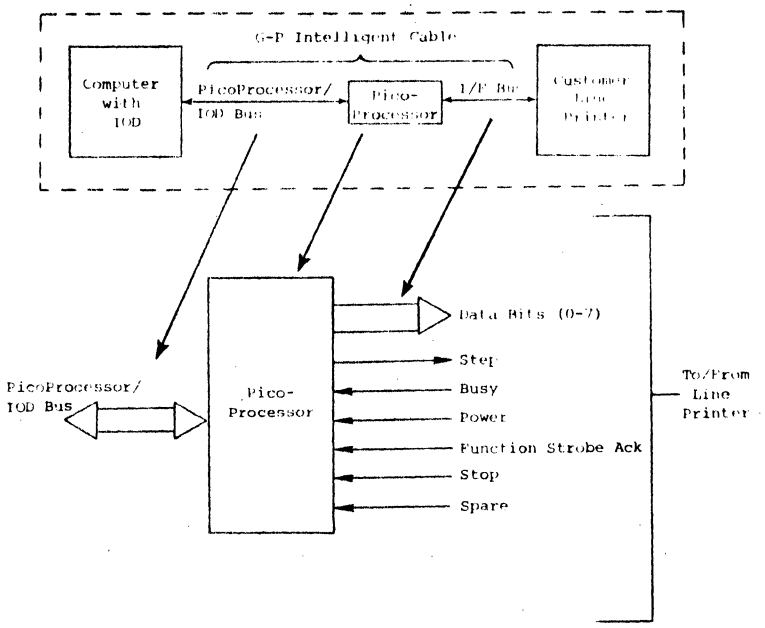



Figure 3-49. Typical Line Printer Interfacing

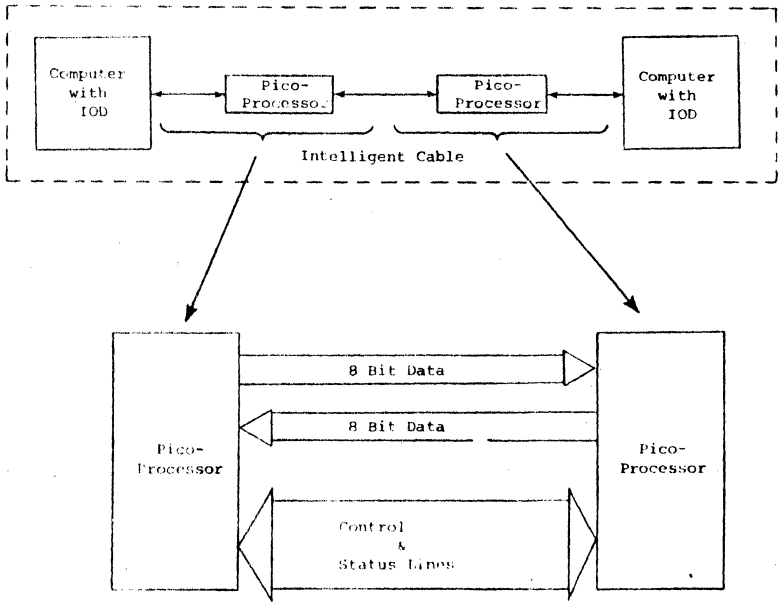



Figure 3-50. Computer-to-Computer Interfacing

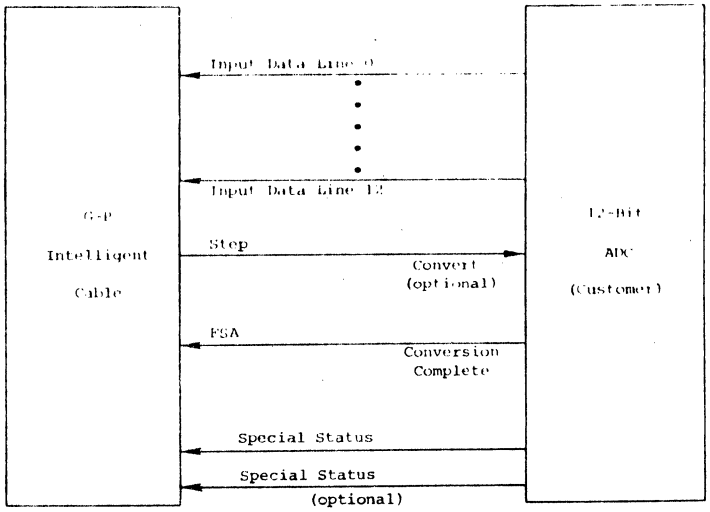



Figure 3-51. Typical Analog-to-Digital Conversion

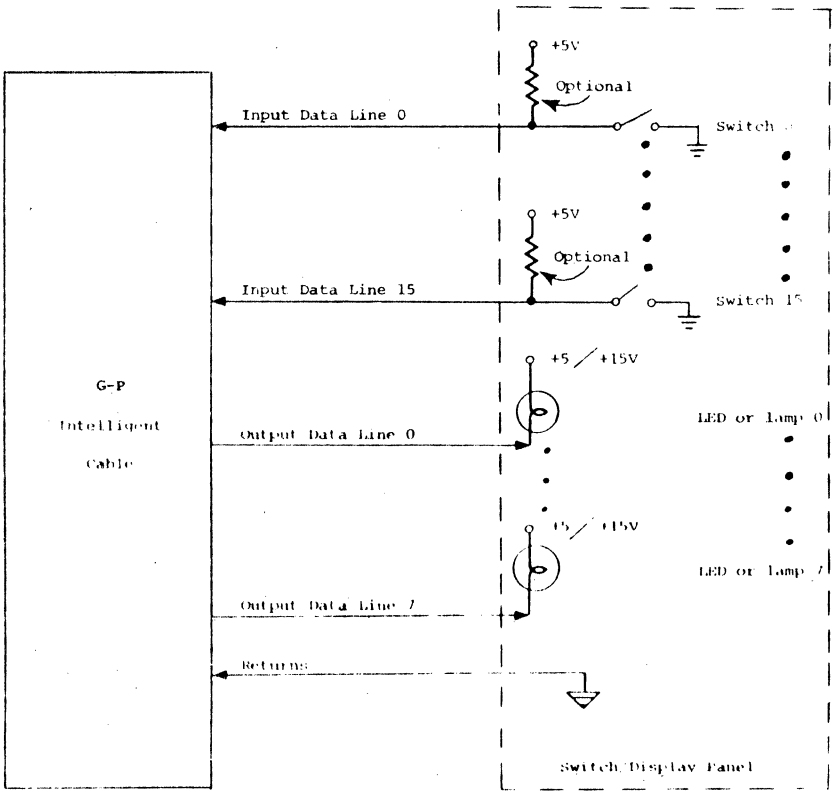



Figure 3-52. Typical Switch/Display Application
(Negative true shown)

3.3.7.7.1 Control Lines (to device)

1. Data/Command. The Data/Command line is driven by the PicoProcessor to indicate either a data or command operation to be performed on the input or output data lines. This line is a logical one for a data transfer and zero to indicate a command operation. This line is used in conjunction with the Input/Output line (defined next).

   The Data/Command line must be established (CPU Command Instruction Word, bit 2) prior to starting an operation and must not be changed during a sequence. The Data/Command line is stable at least 1 sec before the Step or Function Strobe line (described in paragraphs 3 and 5) is asserted (logical one) and remains stable for the entire transfer sequence. During Reset, this line is reset to the Command state (with Input/Output reset to Output).

2. Input/Output. The Input/Output line is driven by the PicoProcessor and used in conjunction with Data/Command to indicate to the device the specific type of operation to be performed. This line is a logical zero for an output operation and one for an input. The specific operation selected by the states of Data/Command and Input/Output lines were given in 3.3.7.3.1.

   The Input/Output line must be established (CPU Command Instruction Word, bit 3) prior to starting an operation and must not be changed during a sequence. The line is stable for at least 1 sec before Step or Function Strobe is asserted (logical 1) and remains stable for the entire transfer sequence. During Reset, this line is reset to the Output state (with Data/Command reset to Command).

   Many devices may be only simplex output or input devices and only require simple handshake or strobes. These devices do not require commands or input/output indicators and therefore it will not be necessary for these types of devices to monitor the Input/Output and Data/Command lines.

3. Function Strobe. The Function Strobe line is driven by the G-P PicoProcessor as a "handshake" line used for both input and output operations.

   For an output operation, this line is asserted until the Function Strobe Acknowledge status signal is received from the device. The device can sample the data bus whenever the Function Strobe line is a logical one or by clocking on the rising or falling edge of this line.

   For an input operation, this line is driven by the PicoProcessor to signal the device that data has been sampled from the data lines. This line remains at a logical one level until Function Strobe Acknowledge is non-asserted (logical zero).

4. Reset. The Reset line is driven by the PicoProcessor under software control or as a result of depressing the RESET switch on the computer's control panel. This signal is a 250 ns (minimum) pulse which can be used by the peripheral device to reset interface electronics to the initialized conditions. The reset line is a negative true line regardless of the interface polarity. The line is normally high and is driven low when RESET is asserted.

5. Step. The step line is a 500 nsec (minimum) pulse or strobe generated by the PicoProcessor. This line is normally used for devices having strobe or non-handshaking I/O disciplines.

During output operations, this line can be used by the device for strobing data from the output data bus lines. During Input operations, the Step line is used to signal the device to place data on the input data lines.

6. End. The End line is generated by the PicoProcessor for both input and output operations to signal the peripheral device that sequence is being aborted or that the byte count is zero (all data has been transferred). This signal is a 2 μsec (min.) pulse normally driven after the trailing edge of Function Strobe for input and output operations.

3.3.7.7.2 Status Lines (from device). The Status Word (shown in figure 3 42) indicates certain operational and error conditions in the device. Five of the bits are checked individually by PicoProcessor firmware when operating in Auto I/O. However, all six lines are available for interrogation by the CPU. When the devices' PicoProcessor receives an input instruction requesting device status, it immediately transfers the entire Status Word to the CPU. When extended status is used, status is input on the data input lines. See 3.3.7.3.1(4). Following are descriptions of the individual bits of the Status Word.

1. Function Strobe Acknowledge (FSA). This line is driven (logical 1) by the peripheral device to handshake with the Function Strobe control line for all input and output operations. During an output operation, FSA is driven to indicate that the data has been sampled from the output data lines. During an input operation, this line is driven by the device to indicate that data has been placed on the data lines. it is stable and may be sampled by the PicoProcessor. For programmed I/O, the duration of FSA must be sufficient to allow the CPU to input status and check for the assertion of FSA.

   For devices with a Strobe I/O discipline, FSA is used in conjunction with the Step line and FSA can be a pulse of 251 ns (minimum).

2. Two Byte. The Two Byte status line indicates to the PicoProcessor that the device is a one-byte or two-byte device. If the line is a logical one, it is a two-byte device; if zero, it is a one-byte device. This line is not checked by the PicoProcessor during output operations.

3. Stop. Stop is generated by the peripheral device to indicate that a device end has occurred and that no more data should be transferred. A device end includes error conditions that should stop all data transfer. Stop is driven (logical 1) by the device with the same timing as the FSA line.

4. Busy. The Busy line is a Not Ready line generated by the peripheral device. This line is an optional line used by devices which have a combination pulse and handshake I/O discipline. This line, when logical 1, signifies that the peripheral device is busy and cannot immediately accept data during an output operation or cannot supply data during a data input operation. This line is driven at the same time (or timing) as the FSA line for the Busy or Not Ready state. The device can then drive this line to the Not Busy or Ready state any time new data can be accepted (output) or data can be supplied (input). During Auto I/O, the PicoProcessor will wait indefinitely for the Not Busy condition without indicating an error.

5. System Power. This line is driven by the device to indicate that device power is turned on, the device connector is installed, and the device is operational.

It is at the logical one level when power is on. Refer to Table 3-6(a) and (b) for connection requirements if the peripheral device cannot supply this status. This line is not checked by the PicoProcessor during input operations but is available to the CPU for software interrogation.

6. Spare. The spare line can be driven by the peripheral device for any purpose desired. The PicoProcessor does not examine this line, but the line will be available for interrogation by the CPU upon an input status request instruction. If this line is not used, refer to Tables 3-6(a) and (b) for connection requirements.

3.3.7.7.3 External Voltage Line. An external voltage is supplied on this line by the peripheral device when interfacing output data and control lines to logic levels greater than +5V (see paragraph 3.3.7.2, Output Lines). An external voltage is not required when the output lines interface to +5V logic levels.

3.3.7.7.4 Data Lines. The PicoProcessor has eight parallel output lines (0 through 7, where bit 7 is the most-significant). The lines are stable for a minimum of 500 ns before the Function Strobe and Step lines are asserted. Data is stable for a minimum of 500 ns after the trailing edge of the Function Strobe.

The PicoProcessor has 16 input data lines (0 through 15, where bit 15 is the most-significant). During input operations, the peripheral device must place data on the lines not later than 250 ns after asserting Function Strobe Acknowledge. The data must remain stable until the leading edge of Function Strobe. If Function Strobe Acknowledge is used as a strobe rather than a handshake line, the data must be stable for at least 100 ns sec after the trailing edge of Function Strobe Acknowledge.

Additional details of interface line timing and sequence information and requirements can be obtained from the flow charts and timing diagrams included under "Firmware Sequence", next paragraph.

3.3.7.8 Firmware Sequence

The General-Purpose Intelligent Cable's PicoProcessor contains two general-purpose firmware sequences -- one for input and one for output operations. The firmware automatically controls data transfers by checking device status and asserting control lines to the device. The entire sequence is described in this section, but operation can begin at any point in the sequence, selected by software. Individual interface control lines can be used or not used, depending on the requirements of the peripheral device.

3.3.7.8.1 Auto I/O Output Sequence. Figure 3-53 is a flow chart of the Auto I/O firmware sequence when performing a Command Output or Data Output operation. Figure 3-54 shows the timing of the interface lines. Arrows in the timing diagram correspond to the sequence of operations indicated by the flow chart.

Following is a description of the flow chart with each segment of the sequence identified by name (function) and sequence address. The yes/no decisions refer to the true/false (logical 1/logical 0) state of a line.

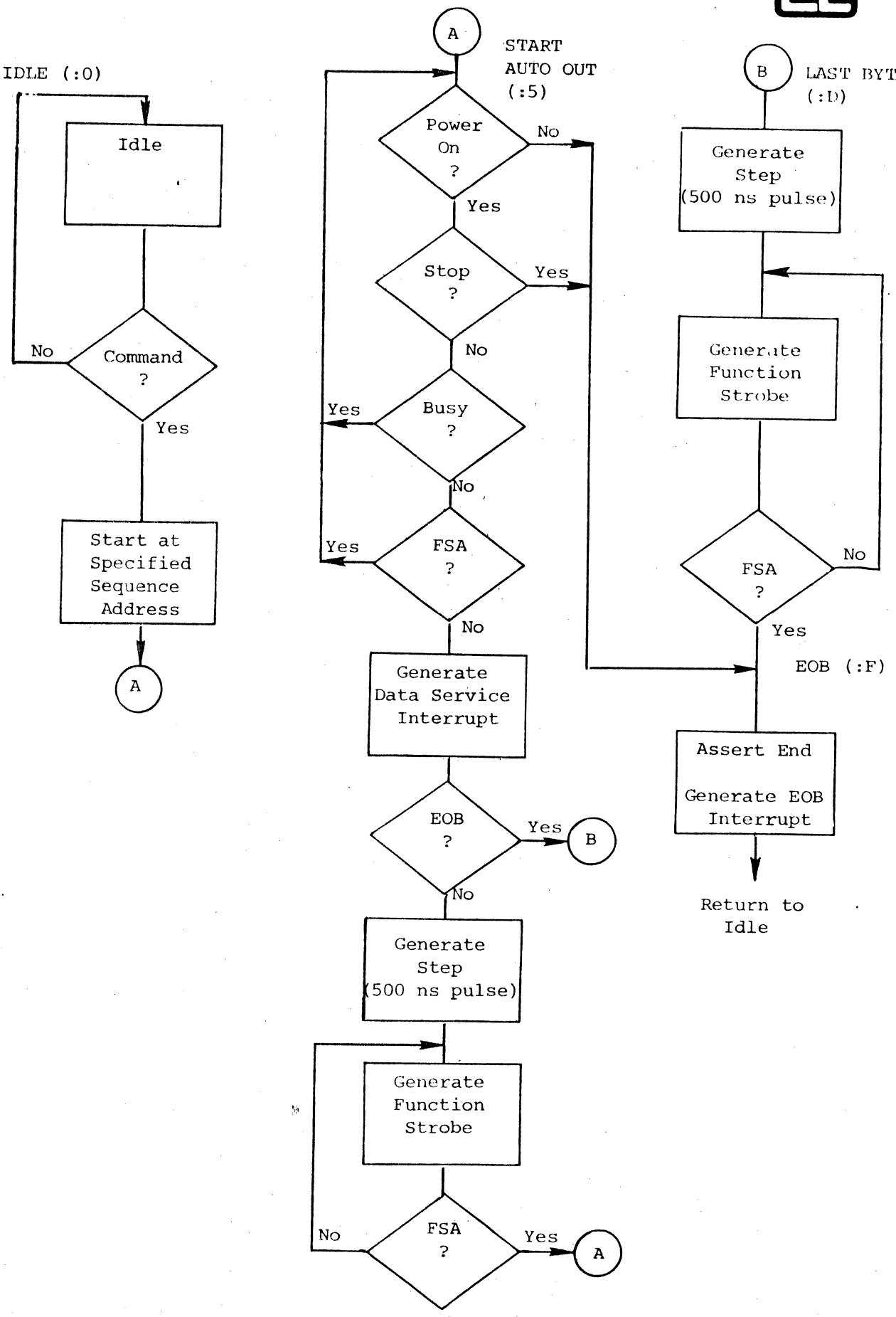
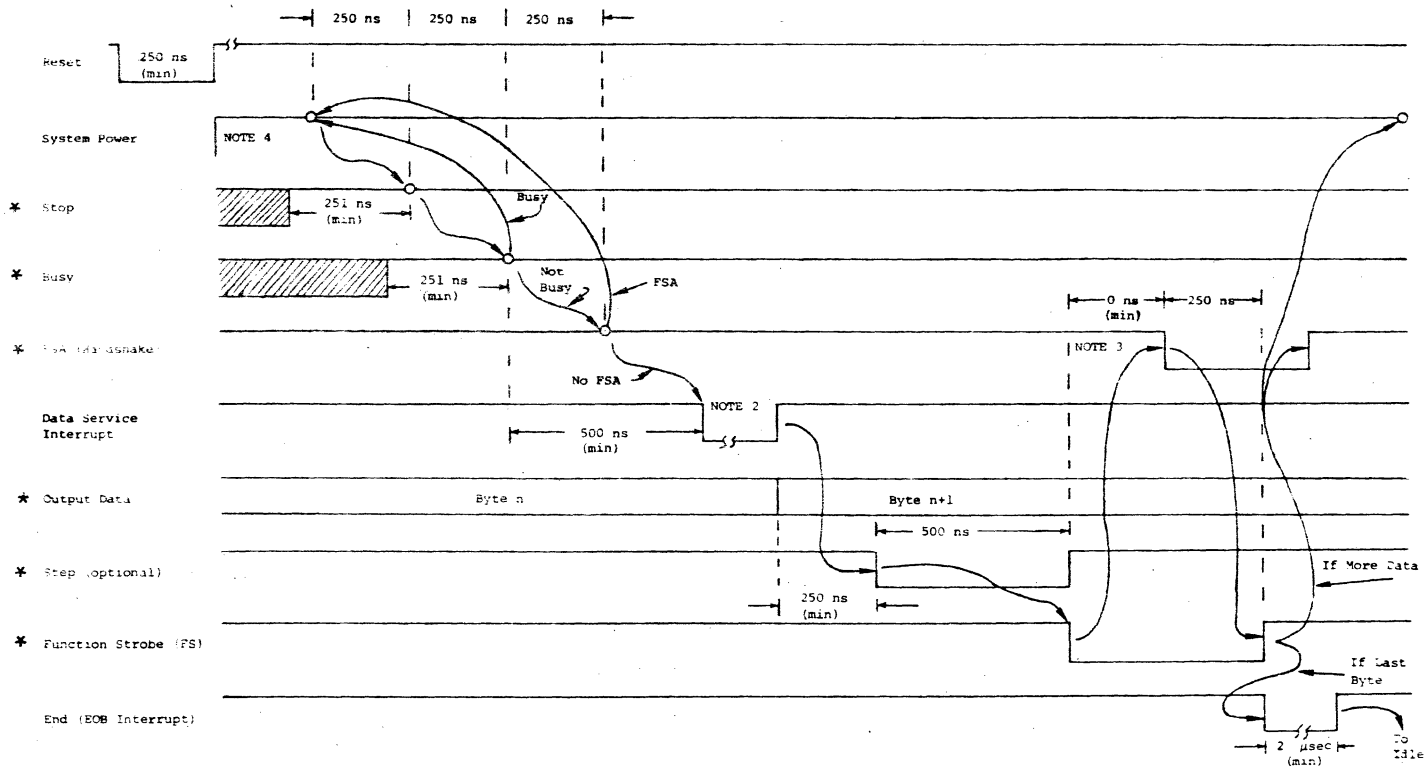



Figure 3-53. Auto I/O Firmware Sequence - G-P Output Operation




NOTE 1: Input/Output and Data/Command lines are not shown, but are stable throughout the sequence.

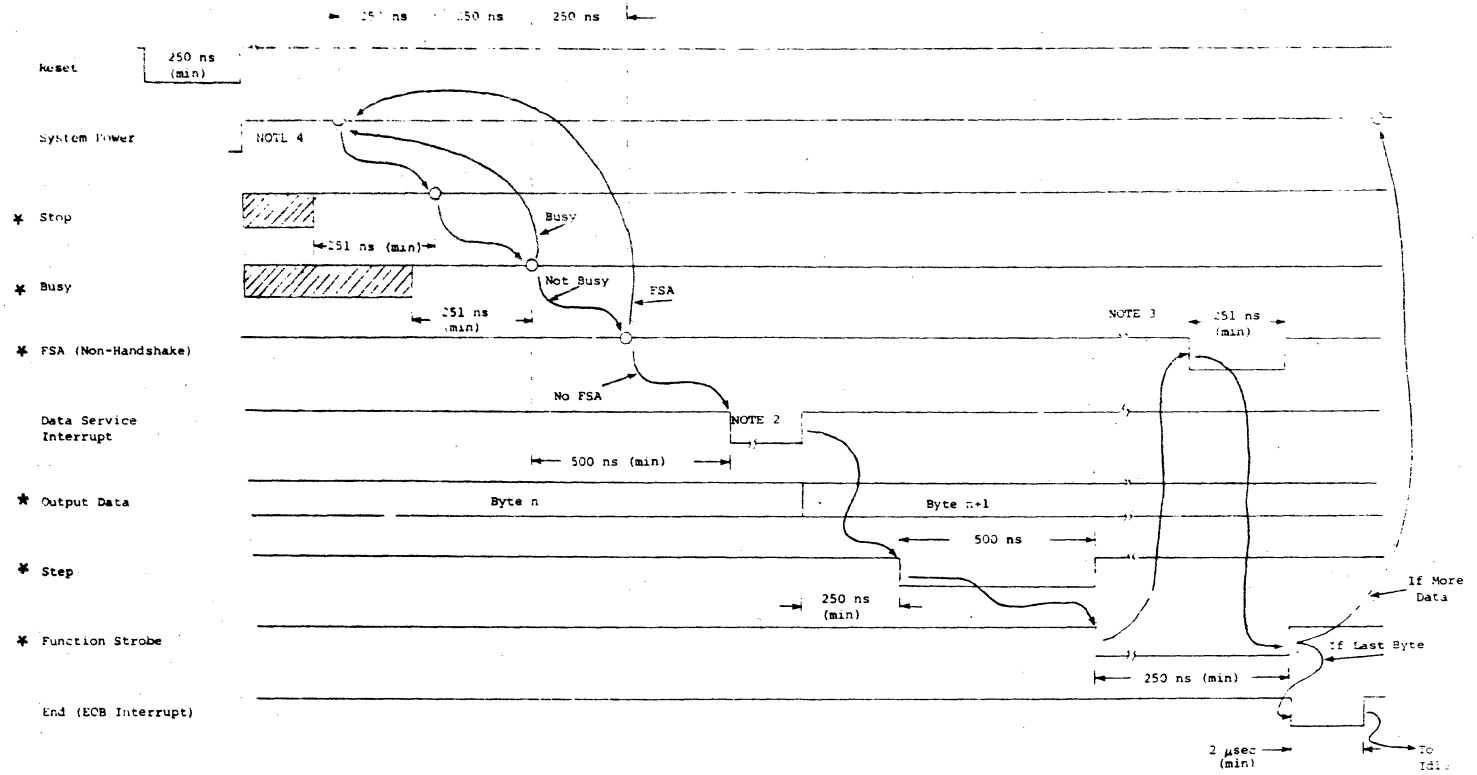
NOTE 2: Minimum time is equal to the instruction time of the Auto I/O byte instruction plus 4 µsec. For instruction times, see the appropriate Computer Handbook.

NOTE 3: The PicoProcessor will wait indefinitely for FSA to be asserted. FS will remain asserted until FSA is received.

NOTE 4: If Power is off or Stop is asserted, the sequence will abort and branch to the EOB sequence.

*: Inverted For Positive True Interface Logic.

Figure 3-54(a). Auto I/O Output Timing For Handshake Interface Discipline (Negative True Shown)




NOTE 1: Input/Output and Data/Command lines are not shown, but are stable throughout the sequence.

NOTE 2: Minimum time is equal to the instruction time of the Auto I/O byte instruction plus 4 µsec. For instruction times, see the appropriate Computer Handbook.

NOTE 3: The PicoProcessor will wait indefinitely for FSA to be asserted. FSA (pulse width = 251 ns, min.) may be asserted 0 ns after the Logical 0 going edge of Step with no maximum restriction. FSA may also be asserted after the Logical 1 going edge of Step if FSA is guaranteed to be at least 500 ns in width.

NOTE 4: If Power is off or Stop is asserted, the sequence will abort and branch to the EOB sequence (assert End and generate EOB interrupt).

*: Inverted For Positive True Interface Logic.

Figure 3-54(b). Auto I/O Output Timing For Non-Handshake Interface Discipline Negative True Shown

IDLE (:0)

Initially, the PicoProcessor is in the Idle state as a result of a Reset or because of a prior End-of-Block interrupt. A Begin command (with mode bit 3 of the Command Word set to zero to select an output operation) causes the PicoProcessor to start a data or command output operation (depending on the setting of mode bit 2) at the sequence address specified by the Branch Address field of the Command Word.

START AUTO OUT (:5)

The PicoProcessor first checks System Power. If this status line is a logical 1 indicating that power is on, the Stop status line is checked. If Stop is false, it indicates that no device error conditions exist and the device has not reached the physical end of its media (tape, card, etc). If an error is indicated by either the Power or Stop status checks, the PicoProcessor immediately generates an End-of-Block interrupt to terminate the transfer. Otherwise, the PicoProcessor checks Busy to verify that the device is ready to accept data. If Busy is true (device not ready), the PicoProcessor returns to the start of the sequence beginning with the Power status check. It repeats the sequence until Busy goes false (device ready). The PicoProcessor then checks the Function Strobe Acknowledge (FSA) status line and waits for the line to go false (logical 0) by repeating the start sequence until it becomes false (FSA may be true from a prior data transfer).

When FSA is false, the PicoProcessor generates a Data Service interrupt. The Auto I/O instruction at the interrupt location is executed and the PicoProcessor copies the CPU data into its output register and drives the output data lines. If the data is the last to be transferred, the PicoProcessor proceeds to the Last Byte sequence.

If more data is to be transferred, the PicoProcessor generates a 500 ns Step Pulse. Step is usually used by pulse (non-handshaking) devices to strobe data into the device. The PicoProcessor then generates Function Strobe (FS). Function Strobe enters data into devices having handshaking I/O discipline.

After the device is strobed by either Step or Function Strobe to enter data into the device, the PicoProcessor waits for the device to signal that it has accepted the data by driving the FSA status line. When FSA goes true, the PicoProcessor returns to the beginning of the Start Auto Out sequence where it checks status and generates another Data Service interrupt to output another byte of data. This continues until the CPU signals the PicoProcessor that all data has been transferred. The Pico-Processor then enters the Last Byte sequence.

LAST BYTE (:D)

The Last Byte sequence consists of the PicoProcessor strobing data into the device by generating Step and Function Strobe as for any other byte. It then waits for the device to acknowledge the data (FSA), then enters the EOB sequence.

EOB (:F)

This sequence is entered when a device status (Power or Stop) error exists or when all data has been transferred from the CPU and to the device. The PicoProcessor generates the End Pulse for those devices that need to know that all data has been transferred while generating an End-of-Block interrupt. The PicoProcessor then returns to the Idle state.

3.3.7.8.2 Input Sequence. Figure 3-55 is a flow chart of the Auto I/O firmware sequence when performing a data input or extended status input operation. Figure 3-56 (a and b) shows the timing of the interface lines. Arrows in the timing diagram correspond to the sequence of operations indicated by the flow chart.

Following is a description of the input flow chart with each segment of the sequences identified by name (function) and sequence address. The yes/no decisions in the flow chart refer to the true/false (logical 1/logical 0) state of a line.

IDLE (:0)

Initially, the PicoProcessor is in the Idle state as the result of a Reset or because of an End-of-Block interrupt. A Begin command (with mode bit 3 of the Command Word set to "1" to select an input operation) causes the PicoProcessor to start an input data or input command (extended status) operation at the sequence address specified by the Branch Address field of the Command Word.

START AUTO IN (:6)

The PicoProcessor first checks the Stop status line. If this line is true (logical 1) indicating a device error, the PicoProcessor generates an End-of-Block interrupt. If the line is false (logical 0), the PicoProcessor generates a 500 ns Step pulse to signal the device to place data on the input data lines. The PicoProcessor then checks the Busy status line to verify that the device is ready to output data. If the line is true (device not ready), the PicoProcessor returns to the start of the sequence (starting with the Stop status check) and repeats the sequence until Busy is false (logical 0) indicating that the device is ready.

The PicoProcessor then waits for FSA to be asserted and then checks the Two-Byte status line. The state of this line determines whether one or two Data Service interrupts are to be generated. If this line is true (logical 1) it indicates to the PicoProcessor that it is a two-byte (16-bit) device. The PicoProcessor then generates a Data Service Interrupt to transfer the first (most-significant) byte. If the transfer was not the last byte to be transferred, the PicoProcessor then generates a second Data Service Interrupt to transfer the second byte of data. If all data has been transferred after the first transfer, the PicoProcessor generates Function Strobe (FS), waits for FSA to go false, and then enters the EOB sequence. If the Two-Byte line was false indicating a one-byte (eight-bit) device, only one Data Service interrupt is generated.

If all data was transferred after the second Data Service interrupt, the PicoProcessor drives the Function Strobe Line (FS) to indicate to the device that data has been sampled from the data bus. Function Strobe is asserted until FSA is dropped (logical 0), the PicoProcessor then proceeds to the EOB sequence.

If All data was not transferred after the Data Service interrupt, the PicoProcessor asserts the Function Strobe line until FSA is dropped, then returns to the start of the Start Auto In sequence.

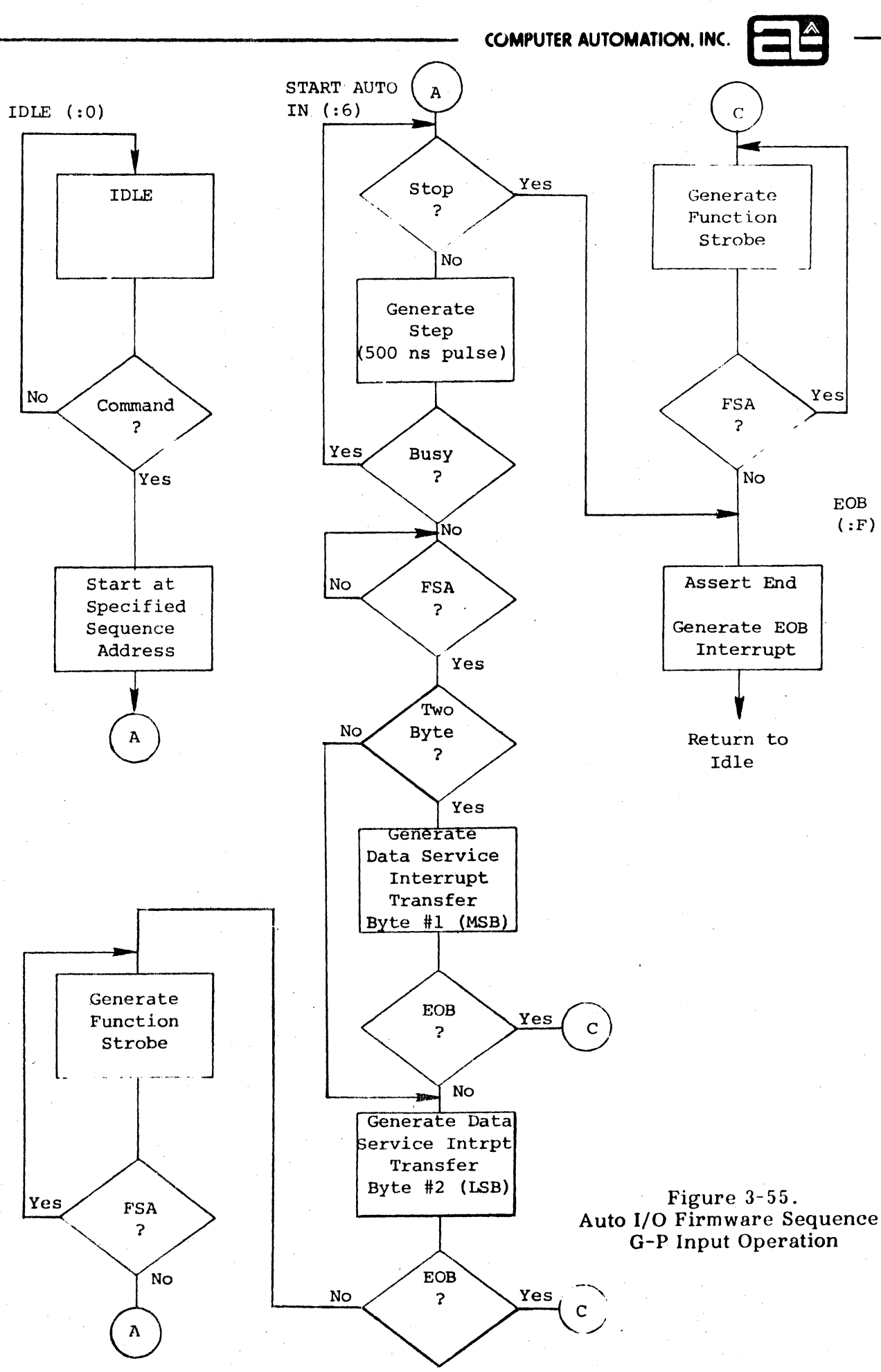
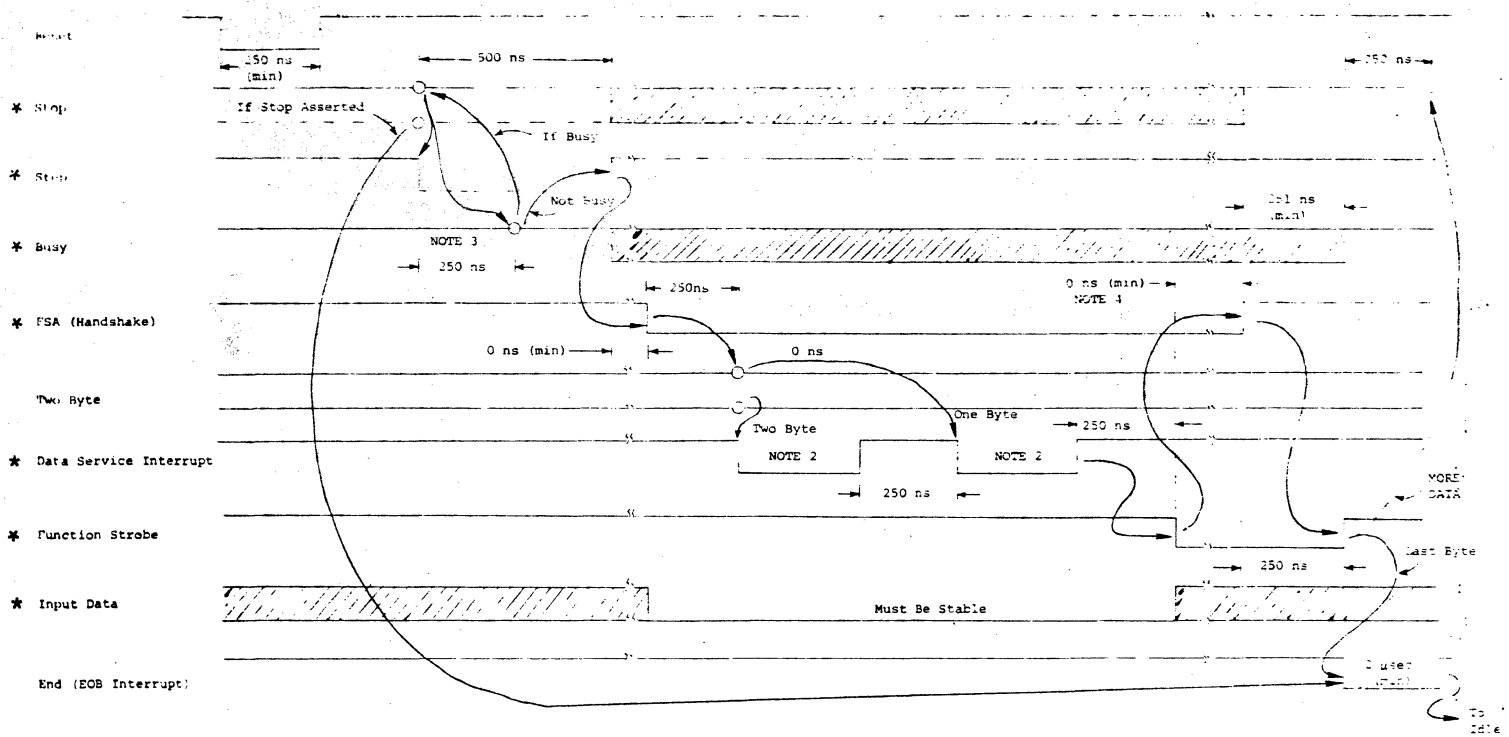



Figure 3-55.
Auto I/O Firmware Sequence,
G-P Input Operation




NOTE 1: The Input/Output and Data/Command lines are not shown, but are stable for the entire sequence.

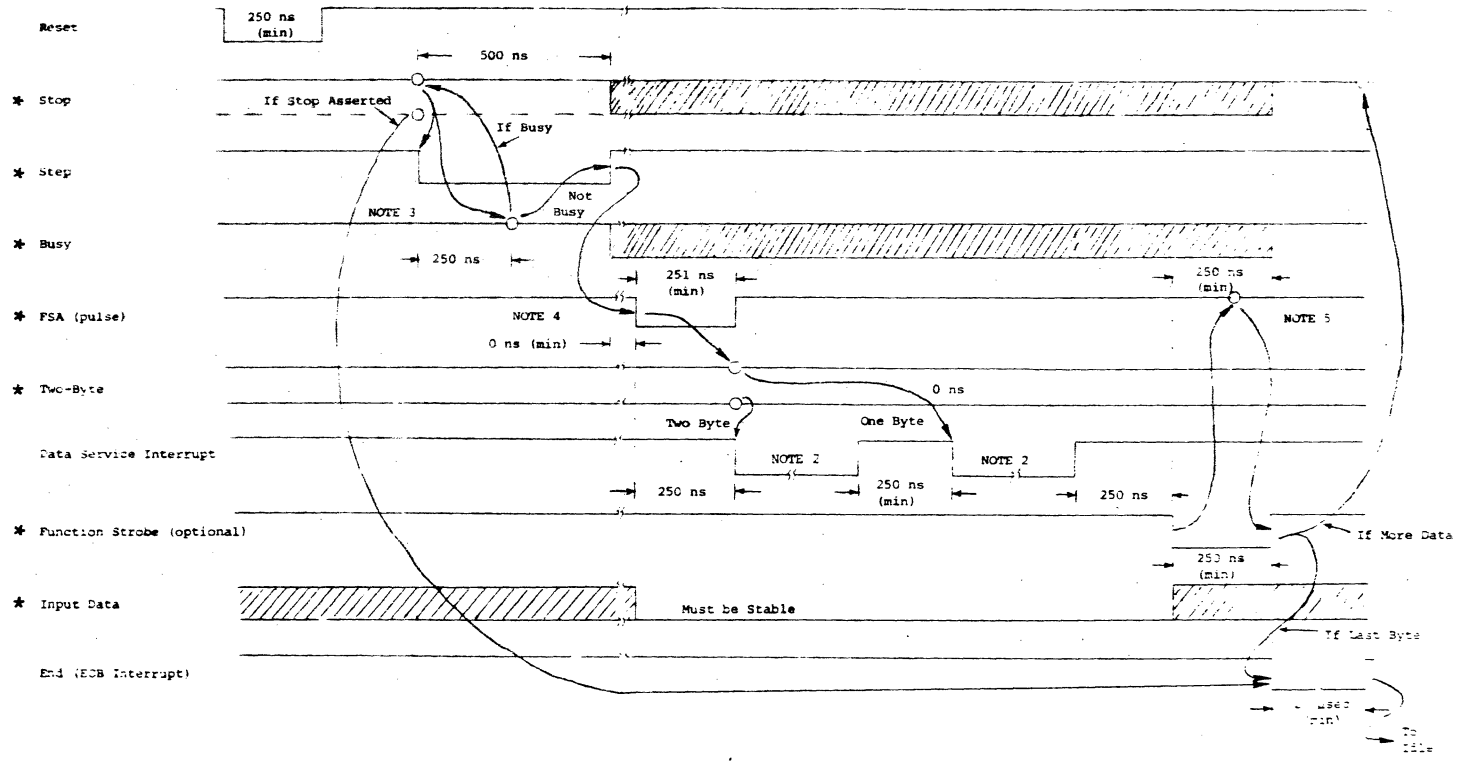
NOTE 2: Minimum time is equal to the execution time of the Auto I/O byte instruction plus 4 μsec. See appropriate Computer Handbook for instruction times.

NOTE 3: If Busy is asserted, the sequence loops back to the start (check Stop). Step remains asserted until the device is Not Busy (ready).

NOTE 4: The PicoProcessor waits indefinitely for the Logical 0 going edge of FSA.

*: Inverted for positive true interface logic.

Figure 3-56(a). Auto I/O Input Timing For Handshake Interface Discipline (Negative true shown)




NOTE 1: The Input/Output and Data/Command lines are not shown, but remain stable for the entire sequence.

NOTE 2: Minimum time is equal to the execution time of the Auto I/O byte instruction plus 4 μsec. See the appropriate Computer Handbook for instruction times.

NOTE 3: If Busy is asserted, the sequence loops back to the start (check Stop). Step remains asserted until the device is Not Busy (ready).

NOTE 4: The PicoProcessor waits indefinitely for the Logical 1 going edge of FSA.

NOTE 5: The PicoProcessor waits indefinitely for FSA to be non-asserted.

*: Inverted for positive true interface logic.

Figure 3-56(b). Auto I/O Input Timing For Non-Handshake Interface Discipline (Negative true shown)

EOB (:F)

The End-of-Block sequence is entered when the FSA line is dropped after the last byte is transferred or when the device drives the Stop line to indicate an error or stop condition. The PicoProcessor then drives the End line for those devices that require notice that all data has been transferred. End-of-Block interrupt is also generated at this time. The PicoProcessor then returns to the Idle state after the EOB interrupt is received by the CPU.

3.3.7.8.3 Programmed I/O. The Programmer can accomplish any desired data, strobe or handshaking sequence or the same sequences shown on the Auto I/O flow charts by use of conventional CPU input/output instructions. However, devices with Strobe I/O disciplines must have strobe widths of sufficient duration to guarantee sampling by the CPU with software routines. The following device control lines and operations can be asserted under software control.

| Programmed I/O Sequence | Mode and Sequence Address | |
|---|---|---|
| | Input | Output |
| Idle or Select Least-Significant Byte (Input) | :0 | :0 |
| End-of-Block Interrupt | :F | :F |
| Function Strobe | :4 | :3 |
| Step (500 ns pulse) | :1 | :1 |
| Select Most-Significant Byte | :3 | -- |
| End | -- | :4 |

Each of the above lines can be asserted by use of a Begin Command causing operation to begin at the point (sequence address) in the firmware sequence where the particular line is asserted. The term "Begin Command" refers to the Begin bit set to "1" and the Branch Address field containing the desired sequence address (see 2.2.4). The Input/Output and Data Command bits (bits 2 and 3 of the mode field in the Command Word) must be set for the desired operation as defined in 3.3.7.3.1.

Following is a description of each Programmed I/O entry sequence listed above.

1. Idle or Select LS Byte (Normally Input Only)

   This sequence (:0) selects input data lines 0 through 7 for input. If the Input/Output mode bit is set to "1" and Step is asserted, one-byte input devices cause an automatic internal branch to this sequence address. It is also entered by a begin command to sequence address :0, a Reset or after an EOB interrupt.

2. EOB Interrupt (Input and Output)

   This sequence (:F) is normally used in conjunction with interrupt programming to force execution of the EOB routine. This is accomplished by a Begin Command to location :F. EOB interrupt remains asserted until the interrupt is recognized by the CPU or another Begin Command or Reset is issued. End is also asserted to the device for the duration of the EOB interrupt sequence. The PicoProcessor then returns to Idle (:0).

3. Function Strobes (FS)

(a) Input Operation. FS is asserted by a Begin Command to :4. It remains asserted until another Begin Command is issued.

(b) Output Operation. FS is asserted by a Begin Command to :3. It is also asserted automatically whenever Step (:1) is asserted even if FS is not required by the device. It must be turned off by a Begin Command to another sequence address.

4. Step

(a) Input Operation. Step is selected by a Begin Command to sequence address :1. It is asserted for 500 ns and then returns to the non-asserted state. When inputting from a one-byte device, after step is asserted, the PicoProcessor automatically branches to Idle/Select LS Byte (:0). When inputting from a two-byte device, the PicoProcessor branches to Select MS Byte (:3) and waits for a new Begin Command.

(b) Output Operation. Step is asserted for 500 ns by a Begin Command to :1. When Step is asserted, the PicoProcessor branches automatically to :3 where FS is asserted (see Function Strobe, Output).

5. Select MS Byte (Input Only)

When inputting from a two-byte device, this sequence is entered by a Begin Command to sequence address :3 or automatically when Step (:1) is asserted to select data lines 8 through 15 for input.

6. End (Output Only) (Non-EOB Interrupt Sequence)

End is asserted by a Begin Command to :4. It may be used to inform the output device that an end-of-transfer has occurred. The End line remains asserted until a new Begin Command is received. See "EOB Interrupt" for additional use of the End line.

The flow charts to be used to describe typical input and output operations generally conform to the timing diagrams of the simple I/O transfers shown in figures 3-43 and 3-44. Actual times given in figures 3-54 and 3-56, however, must be increased by the execution times of the instructions required for Programmed I/O operation. Instruction times are listed in the appendix of the appropriate Computer Handbook. Automatic sequences performed by firmware and not requiring software instructions occur every 250 ns.

1. Typical Output Operation Using Programmed I/O. For a data or command output operation (figure 3-57), a status input instruction can be executed and the device status word interrogated by the CPU. The CPU can determine if device error conditions exist and if the device is ready to accept data. A CPU output instruction can then be executed to transfer data from the CPU to the PicoProcessor's output register.

The data must now be strobed into the device. A Begin Command is issued to the device's PicoProcessor specifying the firmware sequence address where Function Strobe (:3) or Step (:1), whichever tag is appropriate for the particular device, is asserted. Note that if sequence address :1 is accessed to assert Step, Function Strobe is asserted automatically. When not required by the device, the Function

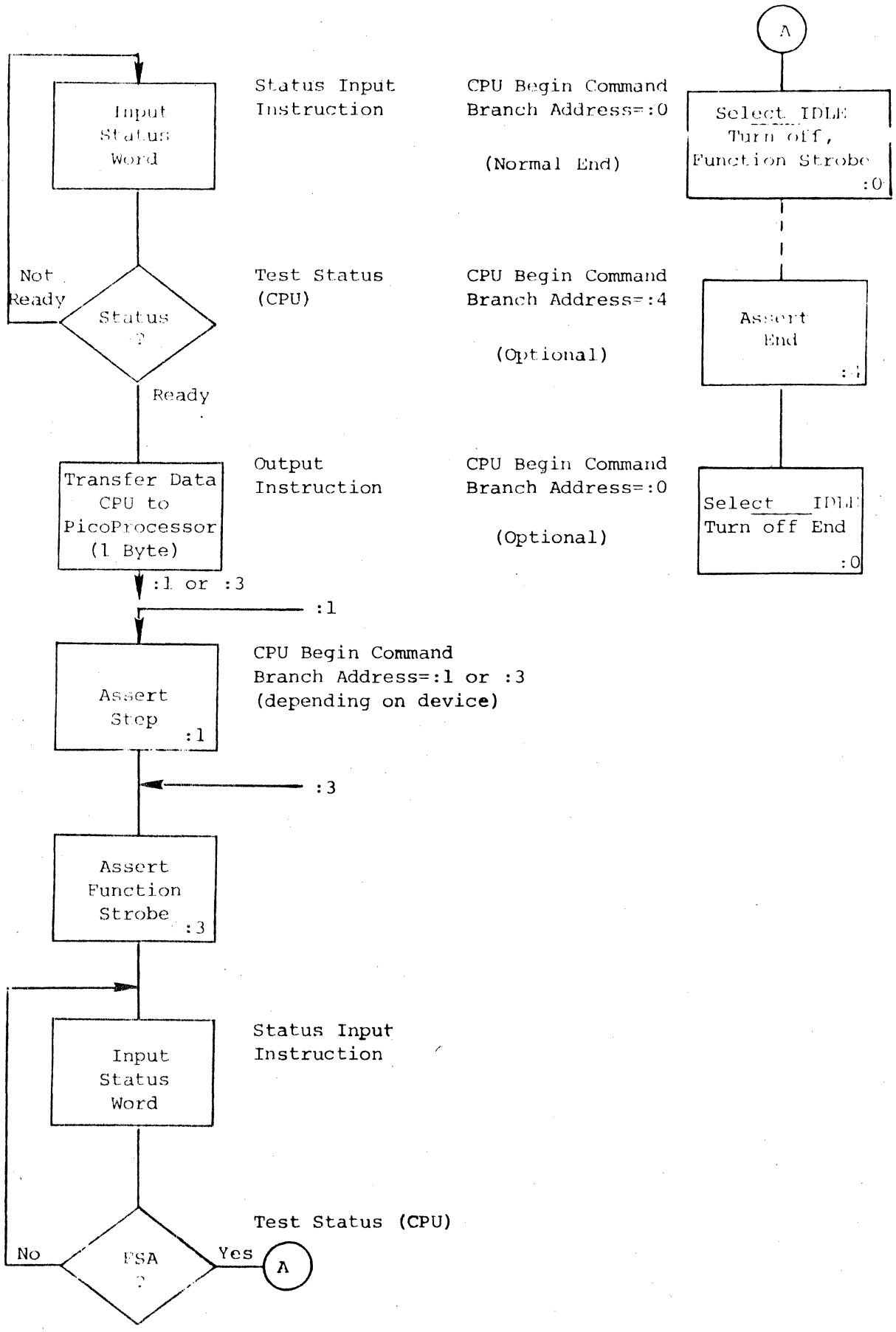



Figure 3-57. Programmed I/O - Typical Output Operation

Strobe line is unused. The CPU can then issue another status input instruction and interrogate the status word to detect when Function Strobe Acknowledge (FSA) is driven by the device (logical 1) to indicate that output data has been accepted. When FSA is asserted by the device, the CPU must issue an Idle (:0) Begin command to drop the assertion of Function Strobe.

Certain devices need to know when the CPU end has occurred or if all data transfers are complete. This can be indicated to the device by the assertion of End by the device's PicoProcessor. This can be done under software by issuing a Begin Command specifying a sequence address of :4. End is also asserted when the end-of-block sequence (:F) is accessed by a Begin Command. This operation resets the device's PicoProcessor to the Idle state after the interrupt is serviced.

2. Typical Input Operation Using Programmed I/O. For a data input or extended status input operation (figure 3-58), a status input instruction can be executed and the device status word interrogated by the CPU. The CPU can determine if device errors exist and when the device is ready to transfer data. The CPU can then issue a Begin Command with the Branch Address field of the Command Word containing the sequence address of :1 to assert Step.

The PicoProcessor automatically branches to the next operation based on the state of the Two Byte status line. If the peripheral device is a two-byte device (16 bits), the Two Byte line is grounded and the PicoProcessor automatically branches to sequence address :3 to select the most-significant byte. The CPU can then detect FSA by executing a status input instruction and interrogating the FSA bit. When FSA is asserted by the device, and input instruction can transfer the most-significant byte to the CPU.

To transfer the next byte (least-significant), a Begin Command is issued accessing sequence address :0. The second byte can be transferred to the CPU by the execution of an input instruction.

When operating with a one-byte device, the first byte transfer (MSB) is skipped and only the least-significant byte is transferred.

When all bytes (one or two) have been transferred, a CPU Begin Command can be issued accessing sequence address :4 to assert Function Strobe. The CPU can execute a status input instruction to input device status and detect FSA dropping to the non-asserted state. When FSA drops, the transfer is complete. A Begin Command accessing sequence address :0 (Idle) is then executed to return Function Strobe to the non-asserted state. And End-of-Block could then be generated as described for the output sequence.

3.3.3 Programming Example

In the example of programming a device operating through the General Purpose Intelligent Cable, it is assumed that the device is a Line Printer whose characteristics will not enable it to operate with the standard Line Printer Intelligent Cable. Table 3-[illegible] shows a simple program listing of an Auto I/O output operation.

The assembler language statements shown in Table 3-[illegible] are those required for using the Distributed I/O System. The reader is referred to the programming information in Section 4. This is a simple example of programming a device for the demonstration only to provide basic transfer with CPU [illegible]

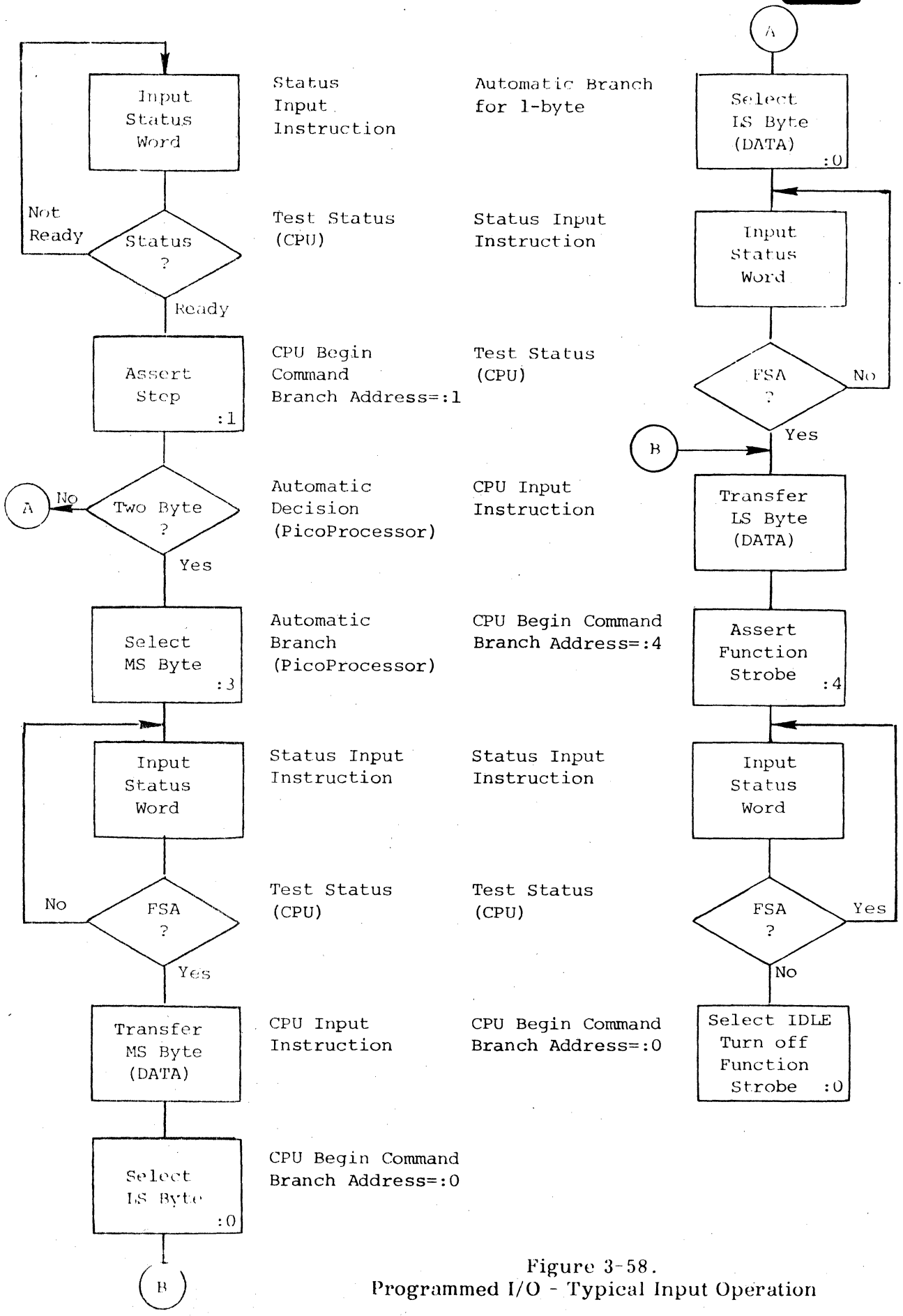



Figure 3-58.
Programmed I/O - Typical Input Operation

Table 3-7. Programming Example - Auto I/O Output, G-P Device

```
GPINT   EQU     (NOTE1)     GENERAL PURPOSE DEVICE INTERRUPT
GPDEVA  EQU     (NOTE2)     GENERAL PURPOSE DEVICE ADDRESS
        ABS     GPINT       INTERRUPT FOR DATA LOCTION
        AOB     GPDEVA      AUTO OUTPUT BYTE
        DATA    $-$         TO BE FILLED WITH BYTE COUNT
        DATA    $-$         TO BE FILLED WITH BUFFER ADDRESS-1
        DATA    A
*
*                           INTERRUPT FOR END-OF-BLOCK
*
        JST     *$+1        CALL END-OF-BLOCK ROUTINE
        DATA    EOB
*
GPRINT  ENI                 ENTRY POINT FOR GP DRIVER
        LDA     BYTCNT      BYTE COUNT FOR MESSAGE
        NAR                 AOB INSTRUCTION NEEDS NEGATIVE
        STA     GPINT+1     PUT IN AOB INSTRUCTION
        LDA     BUFADD      ADDRESS (WORD) OF BUFFER
        LLA     1           AOB INSTRUCTION NEEDS BYTE ADDRESS
        SAI     1           STARTS AT -1
        STA     GPINT+2     PUT IN AOB INSTRUCTION
        LDA     (NOTE3)     SET UP MODE BITS
        OTA     GPDEVA+1    SEND COMMAND TO PICOPROCESSOR
        LDA     (NOTE4)     WORD TO START PICOPROCESSOR
        OTA     GPDEVA+1    START COMMAND TO PICOPROCESSOR
        EIN                 ENABLE INTERRUPTS
        JMP     $           WAIT FOR END-OF-BLOCK
*
*
EOB     ENI                 END-OF-BLOCK INTERRUPT SUBROUTINE
        INA     GPDEVA+1    INPUT STATUS
        RTN     GPRINT      RETURN TO CALLER WITH STATUS IN REG A
```

NOTE 1: Depends on IOD channel used; see figure 2-5, page 2-6.
NOTE 2: Depends on IOD channel used; see table 4-1, page 4-5.
NOTE 3: Mode Field:
- Output Data = :4 (:04x4)
- Output Command = :0 (:04x0)
- Input Data = :C (:04xC)
- Input Extended Status = :8 (:04x8)

NOTE 4: Branch Address Field:
- Output = :5 (:025x)
- Input = :6 (:026x)

Mode and Branch commands can be given at the same time if the user is not switching between input and output sequences.

a dead loop is issued -- JMP$ -- until the PicoProcessor interrupts not to the Data Service interrupt location, but to the End-of-Block location. At this point, control passes to the subroutine labeled EOB.

The demonstration code falls into three distinct parts: Interrupt Locations, I/O Initiation and End-of-Block Service.

1. Interrupt Locations

The first part of the code defines and fills the standard locations associated with IOD channel to which the device's PicoProcessor is connected. As shown in figure 2-4, there are six locations involved of which two -- the Byte Count and the Buffer Location --are dynamic for each I/O Initiation, while the remaining words can be fixed at program load time.

2. I/O Initiation

The second part of the code accomaplishes the transfer of one physical record each time the user's main-line program executes this instruction:

```
CALL GPINT
```

It is assumed that the calling program has previously set up these two words:

```
BYTCNT    Number of bytes to be transferred
BUFADD    Word address of record buffer
```

The demonstration code converts this information into the form required by an Automatic I/O instruction: negative byte count and byte address of the buffer minus one. Once these computations have been stored into the interrupt locations, the PicoProcessor is sent two commands specifying, first, the mode and then the starting Branch Address.

```
Set mode bits (See Note 3, Table 3-7)
Begin at Branch Address :5
```

Notice that the Device Address used in this sequence is coded with the same Device Address used in the interrupt location, plus one. The assembled object code will indicate the transfer of a command, rather than of data, as shown in Figure 2-3.

3. End-of-Block Service

The End-of-Block subroutine could perform error analysis, retry, etc. The demonstration code simply obtains device status from the PicoProcessor and passes status back to the main-line program through the A register.

3.3.7.10 Device Cable Description

The device cable is made up of a 48-conductor flat ribbon cable. It is 1-1/2 feet long and supplied without a connector on the device end. Figure 3-59 identifies each line in the cable. Unused lines must be grounded, left open, or tied to +5V through a pullup resistor as specified in tables 3-6(a) or (b). All ground lines should be used and grounded if possible.

If necessary to remove cable from the PicoProcessor for connector installation, remove the four retaining screws and nuts to remove PicoProcessor cover, then disconnect cable. When replacing cover, take care that the cable is preformed to the shape required to pass under the cover edge without strain. This will assure that firm contact between connectors will not be endangered.

| Conn | Pin | Wire No. | Description | Conn | Pin | Wire No. | Description |
|---|---|---|---|---|---|---|---|
| J4 | 1 | 15 | Stop | J5 | 9 | 18 | Ground |
| | 2 | 13 | Busy | | 10 | 20 | Ground |
| | 3 | 11 | Function Strobe Acknldg. | | 11 | 22 | Data Out 01 |
| | 4 | 9 | System Power | | 12 | 24 | Data Out 03 |
| | 5 | 7 | Reset | | 13 | 26 | Data Out 05 |
| | 6 | 5 | End | | 14 | 28 | Data Out 07 |
| | 7 | 3 | Step | | 15 | 30 | Input/Output |
| | 8 | 1 | Function Strobe (FS) | J5 | 16 | 32 | Data In 01 |
| | 9 | 2 | Ground | J6 | 1 | 47 | Ground |
| | 10 | 4 | Ground | | 2 | 45 | Data In 14 |
| | 11 | 6 | Ground | | 3 | 43 | Data In 12 |
| | 12 | 8 | Ground | | 4 | 41 | Data In 10 |
| | 13 | 10 | Ground | | 5 | 39 | Data In 08 |
| | 14 | 12 | Ground | | 6 | 37 | Data In 06 |
| | 15 | 14 | Ground | | 7 | 35 | Data In 04 |
| J4 | 16 | 16 | Ground | | 8 | 33 | Data In 02 |
| J5 | 1 | 31 | Data In 00 | | 9 | 34 | Data In 03 |
| | 2 | 29 | External Voltage* | | 10 | 36 | Data In 05 |
| | 3 | 27 | Data Out 06 | | 11 | 38 | Data In 07 |
| | 4 | 25 | Data Out 04 | | 12 | 40 | Data In 09 |
| | 5 | 23 | Data Out 02 | | 13 | 42 | Data In 11 |
| | 6 | 21 | Data Out 00 | | 14 | 44 | Data In 13 |
| | 7 | 19 | Spare Status Line | | 15 | 46 | Data In 15 |
| J5 | 8 | 17 | Two-Byte | J6 | 16 | 48 | Data/Command |

*See paragraph 3.3.7.7.3.

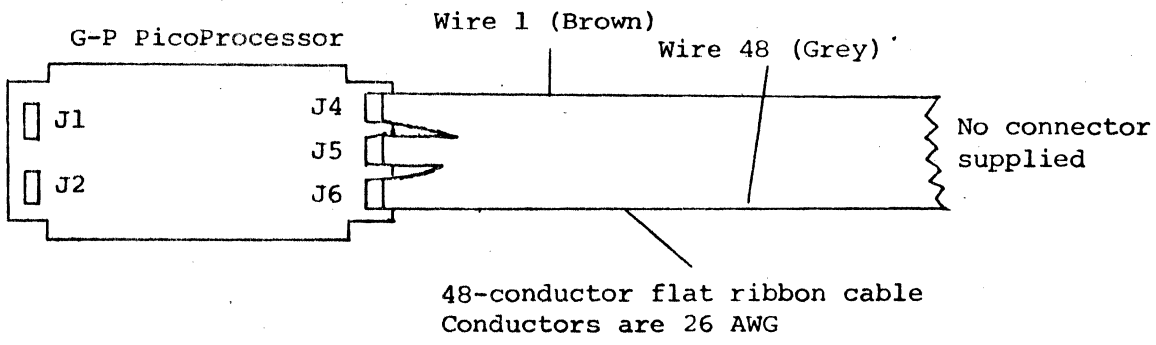



Figure 3-59. Device Cable - G-P Intelligent Cable

## 3.3.8 USER'S MICROCODED INTELLIGENT CABLES

### 3.3.8.1 General Description

The User's Microcoded Intelligent Cables (UMIC) provides a general purpose hardware interface which can be user microprogrammed to meet individual interface requirements. Standard peripherals not supported by other Intelligent Cables can be easily interfaced with an LSI Series Computer using the Distributed I/O System. With the User's Microcoded Intelligent Cable, creation of a unique I/O controller is accomplished by writing a simple microprogram.

The UMIC is supplied as a General Purpose Intelligent Cable, 14631-11, for devices requiring a negative true interface or 14631-12 for devices requiring a positive true interface. The user ROM/PROM set replaces the corresponding standard General Purpose Intelligent Cable ROM's. The sequencing of the Intelligent Cable is then under control of the user coded ROM/PROMs.

### 3.3.8.2 Specifications

**Standard Specifications:** Refer to subsection 3.3.7.2 for the standard specifications of the General-Purpose Intelligent Cables.

**Operating Modes:** Simplex or half duplex, as determined by user microprogram.

**Software Modes:** Programmed I/O or Auto I/O instructions under interrupts, as determined by user microprogram and software requirements.

**Data Modes:** Four modes selectable by software and under control of user microprogram.

1. Data output (8 bits).
2. Data input (8 or 16 bits).
3. Command output (8 bits).
4. Extended status input (16 bits).

**Supporting Software:** Two software packages are available as part of the UMIC Documentation Kit 20631-00, for generation of user ROM/PROM sets:

1. A special set of assembler Macros for assembly of user microprogram.
2. ROM/PROM Format Generator (ROMGEN) Program for generation of user input patterns for procurement of user ROM/PROMs.

**Supporting Documentation:** Two user's manuals are available to aid the user in generation of his custom microprogram (UMIC Doc. Kit 20631-00):

1. User-Microprogrammed Intelligent Cable (UMIC) Reference Manual provides information for coding and assembly of user microprogram.
2. Format Generator (ROMGEN) User's ROM/PROM Program manual provides information to aid the user in generation of ROM/PROM procurement.

### 3.3.9 Magnetic Tape Intelligent Cable

#### 3.3.9.1 General Description

The Magnetic Tape Intelligent Cable (MTIC) 14631-41 controls the transfer of data between a Pertec (or Pertec plug-compatible) magnetic tape formatter and an LSI Family computer.

The formatter unit, in turn, provides interface capabilities for up to four synchronous magnetic tape transports. The various formatters and tape transports available from Pertec include models with combinations of the following capabilities:

- Standard tape speeds from 6.25 to 75 inches per second (IPS)
- Standard recording densities from 200 to 1600 bytes/bits per inch (BPI)
- NRZI or phase-encoded recording formats
- 7- or 9- track configurations
- Read-after-write, read/write, read-only or write-only operation

The MTIC is designed to interface to Pertec formatters that do not have terminating resistor networks on the PICOPROCESSOR (controller) to formatter interface signal lines. Table 3-8 lists several of the Pertec formatters that are compatible with the Magnetic Tape Intelligent Cable. Computer Automation does not provide software support for 7-track or write-only formatters. Other formatters and tape transports may be used with the MTIC if they are fully compatible with the listed Pertec models.

Table 3-8. MTIC-Compatible Pertec Formatters

| Formatter Model Number | Number of Tracks PE | Number of Tracks NRZI | Format Type | Capability | Compatible Pertec Transport |
|---|---|---|---|---|---|
| F829/7 | | 7/9 | NRZI | Read/Write & Read Only | 5X60, 6X60, 7X20, 7X30, 6811, 6812 |
| F849/7 | | 7/9 | NRZI | Read-After-Write | 5X40, 6X40, 7X40 8X40, 8840 |
| F609 | 9 | | PE | Write Only | 5660, 6660, 7620 |
| F619 | 9 | | PE | Read Only | 6611-000 |
| F629 | 9 | | PE | Read/Write | 5660, 6660, 7620 |
| F649 | 9 | | PE | Read-After-Write | 5640, 6640, 7640 8640 |
| F6181 | 9 | 7/9 | PE/NRZI | Read Only | 5612-850, 6612-850 |
| F6282 | 9 | 7/9 | PE/NRZI | Read/Write | 5X60, 6X60, 7X20, 5560, 6660, 7620 |
| F6484 | 9 | 7/9 | PE/NRZI | Read-After-Write | 5X40, 6X40, 7X40, 5640, 6640, 7640 |

3.3.9.1.1 Magnetic Tape System Description

The magnetic tape system interfaced by the MTIC is a two-level, mass data storage peripheral system (Figure 3-60). The individual tape transports provide the basic tape drive mechanism and the read/write circuitry. The tape transports can also sustain and complete a rewind operation once the operation is initiated.

The tape transports operate under the direction of a formatter. Up to four tape transports can be controlled by a formatter, but only one transport may be selected at any given time. Remaining transports may be rewinding, awaiting selection, or turned off. The formatter performs the various "housekeeping" functions necessary to produce a usable tape file. The operations performed by the formatter include:

- Parity generation and checking.
- Check character generation.
- Error correction.
- Inter-record gap timing.

By relieving the CPU of organizing the data to the specific format requirements, the formatter makes possible easier interfacing and programming. For additional information on magnetic tape system operation, refer to the applicable manufacturer's reference documents for the formatter and tape transport.

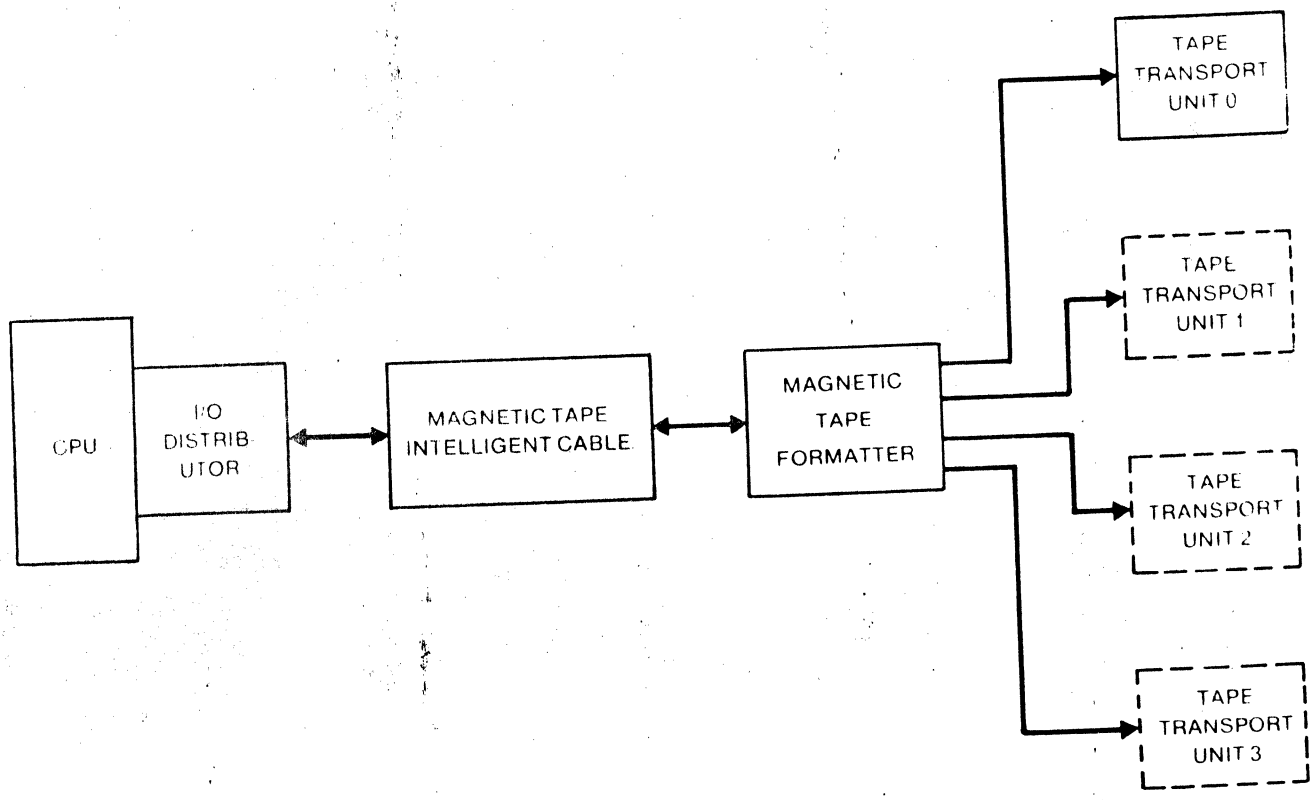



Figure 3-60 Magnetic Tape System Configuration

3.3.9.2 Specifications

Cable length (nominal): 6 ft. (1.8 m)

IOD to PICOPROCESSOR, 4 ft. (1.2 m)
PICOPROCESSOR to formatter, 19 ins (.5 m)

Formatter connector:

CAI paddleboard (female, 100 Pin)

Data types: 8-bit input and output

Tape Compatibility: IBM compatible binary format (i.e. track 0 = most significant bit)

Operating mode: Half-duplex using Auto I/O instructions

Command output: Two command format with 8 mode bits

Status input: Two word 16-bit format

Standard channel number 1 (Device address field :F8)
Standard data service interrupt address :C8
Standard end-of-block interrupt address :CC

Hardware requirements for maximum transfer rates supported:

- For 75 IPS at 1600 BPI

  DMA I/O Distributor, any LSI family computer, any standard memory.*

- For 75 IPS at 800 BPI or 37.5 IPS at 1600 BPI

  Standard I/O Distributor, LSI-2/20 or LSI-2/60 with core 980 memory.*

- For 25 IPS at 1600 BPI

  Standard I/O Distributor, LSI-2/20 or LSI-2/60 with core 1600 or core 1200 memory.*

- For 37.5 IPS at 800 BPI

  Standard I/O Distributor, LSI-2/10 with core 1200 or core 980 memory.*

- For 25 IPS at 800 BPI

  Standard I/O Distributor, LSI-2/10 with core 1600 memory or LSI-3/05 with core 1600, core 1200, or core 980 memory.*

* The requirements listed assumes the computer operation is dedicated to data transfer with all other peripherals in standby condition or powered down. Any attempt to time share operation at the listed maximum rates may produce transfer rate errors.

Strapping on PICOPROCESSOR: none

Strapping on paddleboard: (Figure 3-61)

Select formatter 0 - Standard
Select formatter 1 - FAD to ground

Select odd parity - Standard
Select even parity - PAR to ground

Select high density - Standard
Select high density - DEN to ground

Select low read threshold - Standard
Select high read threshold (not recommended) - THR1 to ground
Select low read threshold (not recommended) - THR2 to ground

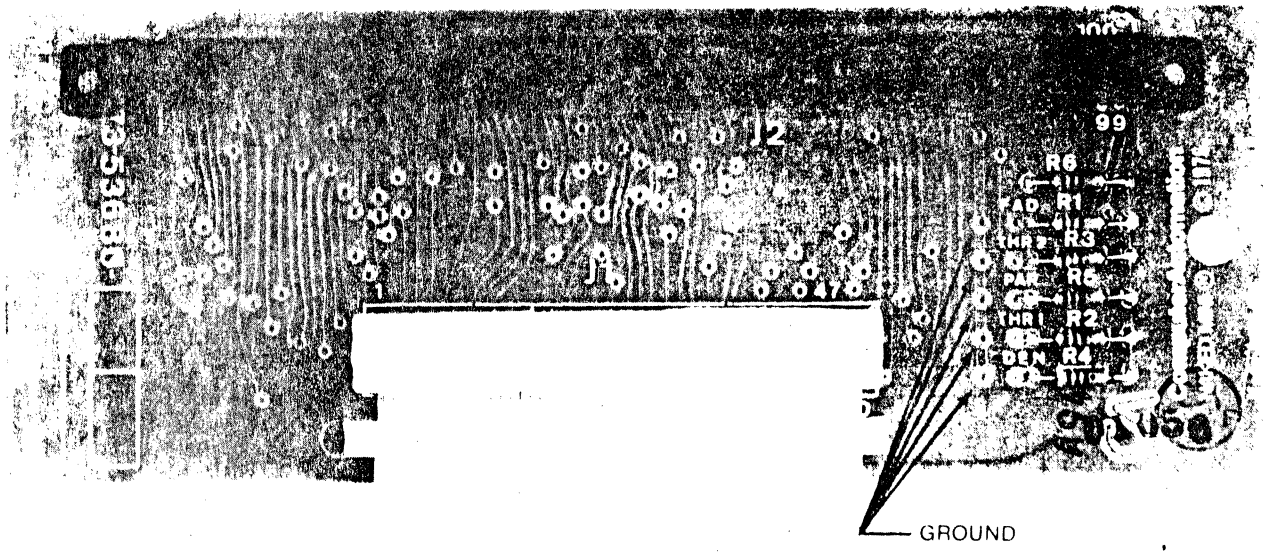



Figure 3-61 MTIC Paddleboard Strapping

3.3.9.3 Software Considerations

The MTIC conforms to the general I/O Distributor programming philosophy for Auto I/O described in Section 2.2. The MTIC PICOPROCESSOR transfers data in 8-bit parallel bytes for both input and output.

The processor command words for both data and control transfer are unchanged from standard usage. The use of extended format for the transfer of both control information to the PICOPROCESSOR and device status to the I/O Distributor requires careful ordering of the command words.

The CPU is able to direct five operational functions of the formatter through the setting of the applicable mode bits. A sixth function is used to command the selected tape transport to offline mode.

The five operational functions are:

- REV - Reverse/forward
- WRT - Write/read
- WFM - Write file mark
- ERA - Erase
- EDIT - Edit feature

These five functions can also be combined to provide more than the five listed functions. For example, the read function can be performed with tape moving in either the forward or reverse direction.

The PICOPROCESSOR firmware sequences are divided into four write sequences and two read sequences. The write sequences are:

- Sense Ready
- Pulse Go
- Rewind /Offline
- Write/Write File Mark

The read sequences are:

- Sense Ready
- Read/Skip

The most significant address bit for the PICOPROCESSOR firmware is derived from the mode register's write/read bit. The four least significant bits are issued as part of the software branch command. Table 3-9 lists the various combinations of the formatter operation functions and firmware sequences used to produce the magnetic tape system functions.

Table 3-9. Magnetic Tape System Functions

| Function | Mode Register Control Bits | | | | | | Firmware Sequence |
|---|---|---|---|---|---|---|---|
| | REV | WRT | WFM | ERA | EDIT | OFL | |
| Write Data | 1 | 0 | 1 | 1 | 1 | 1 | Write :05 |
| Write Data W/Edit | 1 | 0 | 1 | 1 | 0 | 1 | Write :05 |
| Write File Mark | 1 | 0 | 0 | 1 | 1 | 1 | Write :05 |
| Read Data | 0 or 1 | 1 | 1 | X | 1 | 1 | Read :13 |
| Read Data/Edit | 0 | 1 | 1 | X | 0 | 1 | Read :13 |
| Skip 1 Block | 0 or 1 | 1 | 0 | X | 1 | 1 | Read :13 |
| Skip 1 Block/Edit | 0 | 1 | 0 | X | 0 | 1 | Read :13 |
| Rewind | X | 0 | X | X | X | 1 | Rewind :04 |
| Offline | X | 0 | X | X | X | 0 | Rewind :04 |
| Erase (Fixed Length) | 1 | 0 | 0 | 0 | X | 1 | Write :05 |
| Erase (Variable Length) | 1 | 0 | 1 | 0 | X | 1 | Write :05 |
| Sense Ready | X | X | X | X | X | 1 | Sense :01 or :11 |

X = Don't Care

3.3.9.3.1 CONTROL INFORMATION

The format used for the transfer of control information separates the transfer of a branch address from the transfer of the mode register bits. This is required so that the eight mode register bits can be transferred.

The specific magnetic tape operation performed is selected by the configuration of the mode register bits at the time a PICOPROCESSOR firmware sequence is initiated by the CPU. The mode bits also provide the second level addressing for the tape transports and the fifth address bit for the PICOPROCESSOR firmware sequences.

The mode bits must be set before a branch command is issued to start a firmware sequence. The mode bits must not be changed during a firmware sequence.

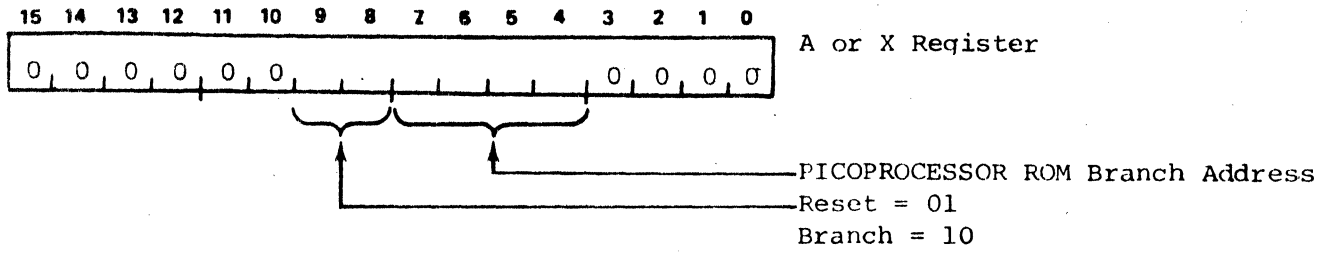
The mode bit assignments are shown in Figure 3-62. The following describes the functional use of the mode bits:

Bit 0 - Transport Address Select 1 (TAD1)

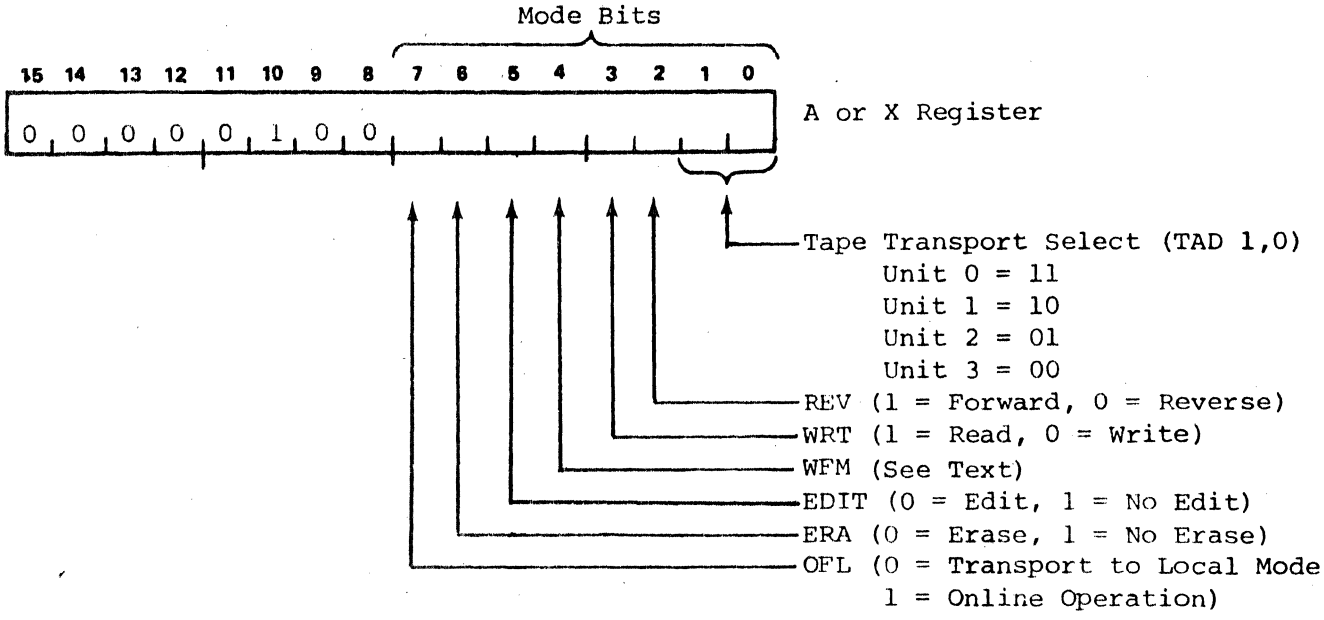
Bit 1 - Transport Address Select 0 (TAD0)

The complementary binary configuration of bits 0 and 1 addresses up to four tape transports attached to the formatter.

Bit 2 - Reverse/Forward (REV). The logical 1 selects forward tape movement in the selected tape transport.




BRANCH OR RESET FORMAT




SET MODE FORMAT

Figure 3-62 Control Information Format

Bit 3 - Write/Read (WRT). The logical 1 selects a read operation in the selected tape transport. Bit 3 also provides the most significant address bit for the PICOPROCESSOR firmware.

Bit 4 - Write File Mark (WFM). For write operations, a logical 0 modifies the write firmware sequence to the write file mark operation; a logical 1 disables the write file mark operation. For read operations, a logical 0 modifies the read firmware sequence to the skip operation; a logical 1 allows the normal read operation.

Bit 5 - Edit. The logical 0 selects the formatter edit function.

Bit 6 - Erase (ERA). The logical 0 selects the formatter erase function.

Bit 7 - Offline (OFL). The logical 0, along with the computer issuing a rewind branch command, places the selected tape transfer in local mode. The logical 1 allows normal online operation.

### 3.3.9.3.2 STATUS INFORMATION

The Magnetic Tape Intelligent Cable's PICOPROCESSOR provides 16 bits of status information for the computer. The MTIC status bits are arranged in two separate status bytes of eight bits each. A single status request from the CPU causes the PICOPROCESSOR to send one status byte (eight bits) to the CPU. The status byte sent may be either status byte 1 or status byte 2. When the CPU releases the status request, the PICOPROCESSOR automatically prepares to send the alternate status byte. A second status request is required to complete the transfer of all 16 status bits.

The order in which the status bytes are transferred, byte 1 or byte 2 first, is determined by the number of status requests made since the most recent reset or mode select operation. The PICOPROCESSOR initializes the status byte control to send status byte 1 anytime a reset or mode select operation is made. The distributed I/O status input operation is described in Section 2.2.5.

The individual status bits are derived from one of the following three sources in the magnetic tape system:

- PICOPROCESSOR
- Formatter
- Tape transport

Status bits reported from the PICOPROCESSOR or formatter represent the present status of the magnetic tape system. However, status bits reported from the tape transport represent the present status of only the selected tape transport. Status for the remaining three (maximum) tape transports is not reported.

The bit assignments for status byte 1 are shown in Figure 3-63. The following describes the magnetic tape system conditions reported by each status bit:

Bit 0 - Error Being Corrected (CER), PE formatters only. The logical 0 indicates formatter is correcting a read error. The computer must issue a reset or mode select to clear a logical 1 on this line.

Bit 1 - PicoPresent (PRE). The logical 0 indicates the PICOPROCESSOR is electrically connected to the I/O Distributor.

Bit 2 - Rewinding (RWD). The logical 0 indicates the selected tape transport is rewinding tape.

Bit 3 - File Mark Indicator (FMK). The logical 0 indicates the formatter detected a filemark. I/O Distributor must assert REST or SELT to clear a logical 1 on this line.

Bit 4 - Load Point (LPT). The logical 0 indicates the tape on the selected tape transport is at the beginning of tape (BOT) marker.

Bit 5 - End of Tape (EOT). The logical 0 indicates the End-of-Tape marker was detected on the selected tape transport. The computer must issue a reset of mode select to clear a logical 1 on this line.

Bit 6 - Check Character Gate (CCG). NRZI formatters only. The logical 0 indicates a check character (CRCC or LRCC) was encountered.

Bit 6 - Identification Burst (ID). PE formatters only. The logical 0 indicates an identification burst was detected.

Bit 7 - NRZ. The logical 0 indicates the selected tape transport is an NRZI unit.

The bit assignments for status byte 2 are shown in Figure 3-63. The following describes the magnetic tape system conditions reported by each status bit.

Bit 0 - Formatter Busy (FBY). The logical 0 indicates that a formatter operation is in progress.

Bit 1 - Data Busy (DBY). The logical 0 indicates the magnetic tape movement is up to speed on the selected tape transport.

Bit 2 - Ready (RDY). The logical 0 indicates the selected tape transport is available for operation.

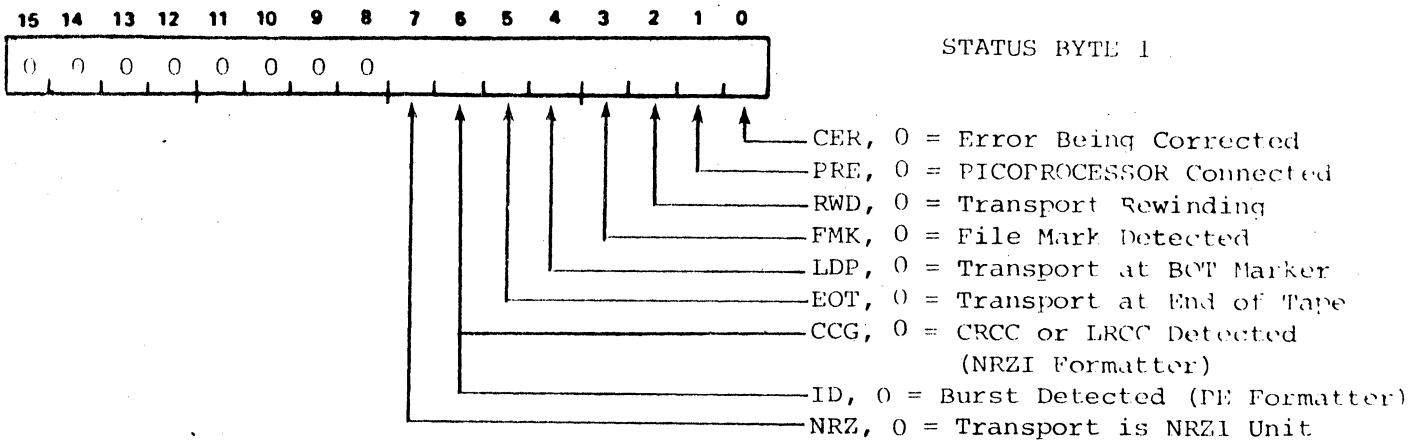
Bit 3 - Character Ready (CHR). During read operations, the logical 0 indicates a read data character from the formatter is ready for transfer to the I/O Distributor. During write operations, the logical 0 indicates the write data buffer can accept another data character.

Bit 4 - On Line (ONL). The logical 0 indicates the selected tape transport is on line.

Bit 5 - File Protect (FPT). The logical 0 indicates the file is protected by the removal of the write enable ring on the supply reel of the selected tape transport.

Bit 6 - Rate Error (RAT). The logical 0 indicates a data service interrupt was not acknowledged soon enough to avoid data loss (Read) or character duplication (Write).

Bit 7 - Hard Error (HER). The logical 0 indicates the formatter detected a non-correctable error. The computer must issue a reset or mode select to clear a logical 0 on this line.



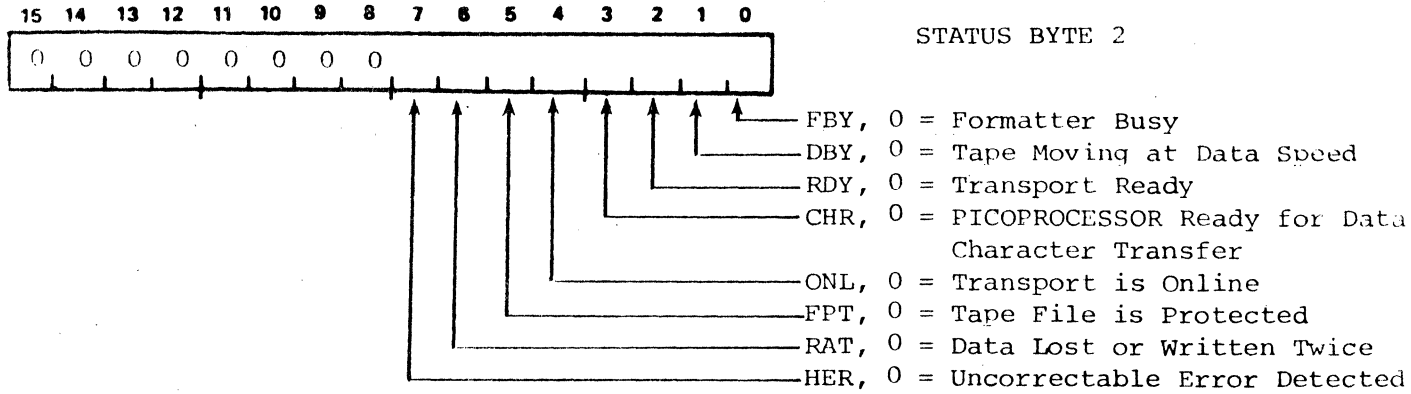




Figure 3-63 Status Byte Formats

### 3.3.9.3.3 SPECIAL CONSIDERATIONS

The special software considerations consist of the following recommended command sequences.

- Do not issue a set mode instruction unless the PICOPROCESSOR is known to be in the idle state.
- Do not issue a set mode instruction that changes tape movement direction or the class of instruction (read to write or write to read) unless the formatter is known to be not busy (FBY=0).
- Always request status, both bytes, following a set mode that changes the tape transport selection.
- Do not issue a branch instruction if the configuration of the mode register is not known.
- Whenever possible, use set mode rather than reset to clear the PICOPROCESSOR.
- Computer Automation recommends that any status request should always be for both status bytes. This reduces the risk of an erroneous interpretation of the status bits.

Refer to Section 3.3.9.4 for further details on overall system operation and Section 3.3.9.5 for a specific programming example.

3.3.9.4 Operating Sequence

The operation of the magnetic tape peripheral system is controlled by software direction of the formatter and tape transport functions together with operation sequencing provided by software selection of the appropriate PICOPROCESSOR firmware sequence.

3.3.9.4.1 Software Sequence

All online operations of the magnetic tape peripheral system are initiated by software issuing a branch command to the PICOPROCESSOR. Prior to the issuance of the command, the software must determine the present conditions of both the formatter and selected tape transport. The software must also select the function to be performed.

Figure 3-64 is a flow chart of the type of software sequence necessary to perform a magnetic tape I/O operation. The following is a description of that software sequence.

DETERMINE INITIAL CONDITIONS

Prior to initiating a magnetic tape operation, it must be determined that an operation is not presently in progress.

**CAUTION**

The PICOPROCESSOR firmware is partially protected since a branch address is not accepted as a branch address by the firmware if the firmware is not in idle but may be accepted as the response to a data service interrupt. The mode bits are not protected and must not be changed during the execution of an operation.

To determine that an operation is not in progress, the software must examine the status bits transferred from the PICOPROCESS. If status was requested and stored at the termination of the previous operation, the software can examine the stored status bits. If an EOB interrupt was expected and not received, an operation is in progress. The status required to allow initiation of an on-the-fly function is PicoPresent (PRF) = 0, Data Busy (DBY) = 1 and Transport Ready (RDY) = 0.

SET THE MODE BITS

The mode bits are set to select the tape transport for operation, and to determine the function being performed. The mode bits must be set prior to execution of a branch command that initiates a firmware sequence. Once set for a particular tape transport and function, additional set mode commands are not required or allowed until either a different transport or function must be selected.

When the next operation involves a change in tape movement direction or changes the class of instruction (read to write or write to read), the set mode instruction must not be issued until the formatter is not busy (FBY = 1).

CHECK DEVICE STATUS

Any time the transport address mode bits (TAD0, TAD1) are changed by the set mode command, status must be interrogated before the branch command is issued. This interrogation is required to assure that the newly-selected tape transport is available for operation. The required status is transport ready (RDY) = 0.

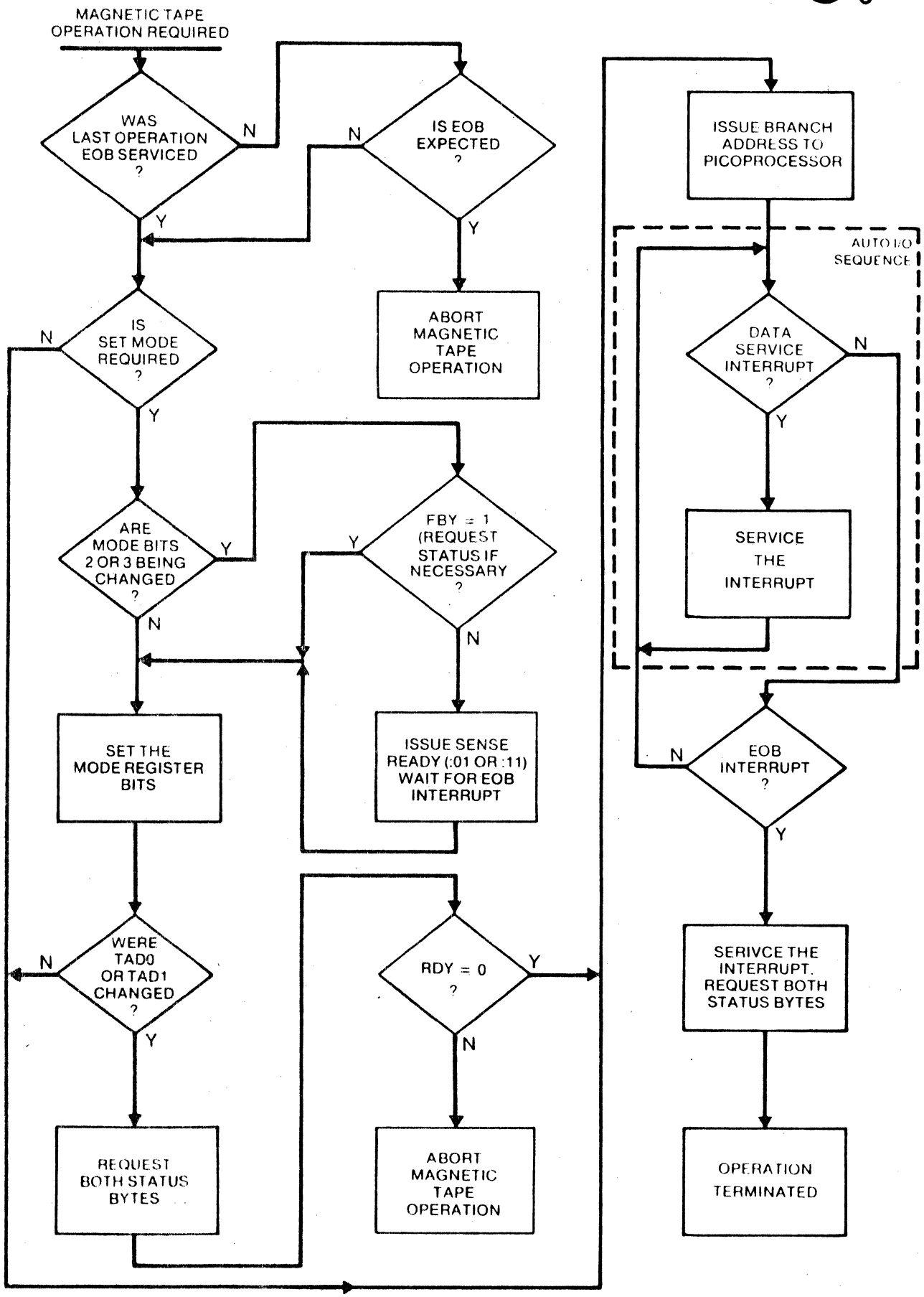



Figure 3-64 Typical Software Sequence

WRITE Data or Filemark (:05)

The Write sequence first checks for the tape transport ready and the tape file not protected (write enable ring installed). If either test fails, the firmware immediately steps to the End-of-Block interrupt and holds the interrupt until acknowledged. If both tests pass, tape motion is initiated. The firmware then checks for the mode register write filemark bit. If WFM = 0, the firmware waits for the formatter to complete the write filemark operation. Write filemark then picks up the firmware sequence at the issuing of the last word tag. If WFM = 1, the firmware immediately issues a data service interrupt for the first data character. When the interrupt is acknowledged, the firmware issues a data service interrupt for the second data character. When the interrupt is acknowledged, the PICOPROCESSOR write data buffer is filled. Subsequent data service interrupts occur each time the formatter signals that a character was written. This continues until the byte count = 0 signal is received from the I/O Distributor. The firmware then waits until the next to the last character is written after which, the last data character is sent to the formatter along with the last word signal. The firmware hold the last word signal until the formatter releases the data busy line. The firmware then steps to the End-of-Block interrupt and holds the interrupt until acknowledged.

READ/SKIP (:13)

The Read/Skip sequence first initiates tape movement, sets a timer (2 seconds nominal), and waits for a data block to be encountered. If a data block is encountered within two seconds, the timer is disabled. If the timer expires or the formatter releases the data busy line without encountering a data block (file mark detection), the firmware steps to the End-of-Block interrupt and holds the interrupt until acknowledged.

NOTE

Tape movement is not stopped if a data block is not encountered within the time allowed. The End-of-Block interrupt is issued after two seconds to allow for software intervention.

When a data block is encountered, the firmware checks the mode register write file mark bit. If WFM = 0, the operation is a skip and data service interrupts are not issued. If WFM = 1, the firmware issues data service interrupts on a per-character basis until the entire block is read or the I/O Distributor issues the byte count = 0 signal. In either case, the firmware does not step to the End-of-Block interrupt until the formatter releases the data busy line signaling the end of the data block. The firmware holds the End-of-Block interrupt until acknowledged.

3.3.9.4.3 Additional Controls

The following are descriptions of the additional magnetic tape system controls available to the software:

RESET

The Reset function is performed by issuing a Reset command. Reset initializes all PICOPROCESSOR logic and resets the formatter. This causes any formatter controlled tape operation to terminate and the transport to stop. Issuing a reset command is the only method of terminating a read forward onto blank tape or read reverse onto the beginning of tape.

WRITE Data or Filemark (:05)

The Write sequence first checks for the tape transport ready and the tape file not protected (write enable ring installed). If either test fails, the firmware immediately steps to the End-of-Block interrupt and holds the interrupt until acknowledged. If both tests pass, tape motion is initiated. The firmware then checks for the mode register write filemark bit. If WFM = 0, the firmware waits for the formatter to complete the write filemark operation. Write filemark then picks up the firmware sequence at the issuing of the last word tag. If WFM = 1, the firmware immediately issues a data service interrupt for the first data character. When the interrupt is acknowledged, the firmware issues a data service interrupt for the second data character. When the interrupt is acknowledged, the PICOPROCESSOR write data buffer is filled. Subsequent data service interrupts occur each time the formatter signals that a character was written. This continues until the byte count = 0 signal is received from the I/O Distributor. The firmware then waits until the next to the last character is written after which, the last data character is sent to the formatter along with the last word signal. The firmware hold the last word signal until the formatter releases the data busy line. The firmware then steps to the End-of-Block interrupt and holds the interrupt until acknowledged.

READ/SKIP (:13)

The Read/Skip sequence first initiates tape movement, sets a timer (2 seconds nominal), and waits for a data block to be encountered. If a data block is encountered within two seconds, the timer is disabled. If the timer expires or the formatter releases the data busy line without encountering a data block (file mark detection), the firmware steps to the End-of-Block interrupt and holds the interrupt until acknowledged.

NOTE

> Tape movement <u>is not</u> stopped if a data block is not encountered within the time allowed. The End-of-Block interrupt is issued after two seconds to allow for software intervention.

When a data block is encountered, the firmware checks the mode register write file mark bit. If WFM = 0, the operation is a skip and data service interrupts are not issued. If WFM = 1, the firmware issues data service interrupts on a per-character basis until the entire block is read or the I/O Distributor issues the byte count = 0 signal. In either case, the firmware does not step to the End-of-Block interrupt until the formatter releases the data busy line signaling the end of the data block. The firmware holds the End-of-Block interrupt until acknowledged.

### 3.3.9.4.3 Additional Controls

The following are descriptions of the additional magnetic tape system controls available to the software:

RESET

The Reset function is performed by issuing a Reset command. Reset initializes all PICOPROCESSOR logic and resets the formatter. This causes any formatter controlled tape operation to terminate and the transport to stop. Issuing a reset command is the only method of terminating a read forward onto blank tape or read reverse onto the beginning of tape.

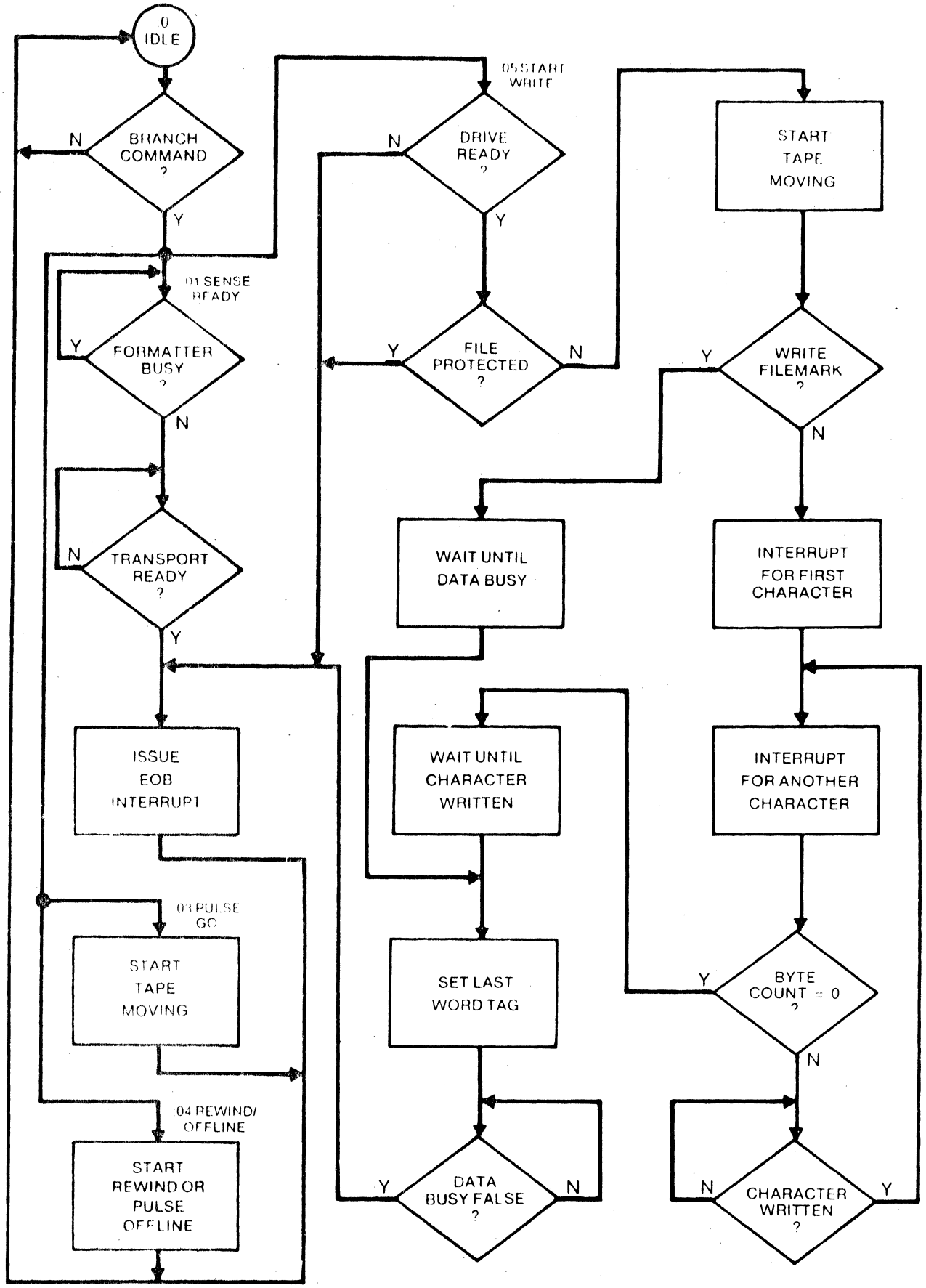



Figure 3-65a Write Firmware Sequence

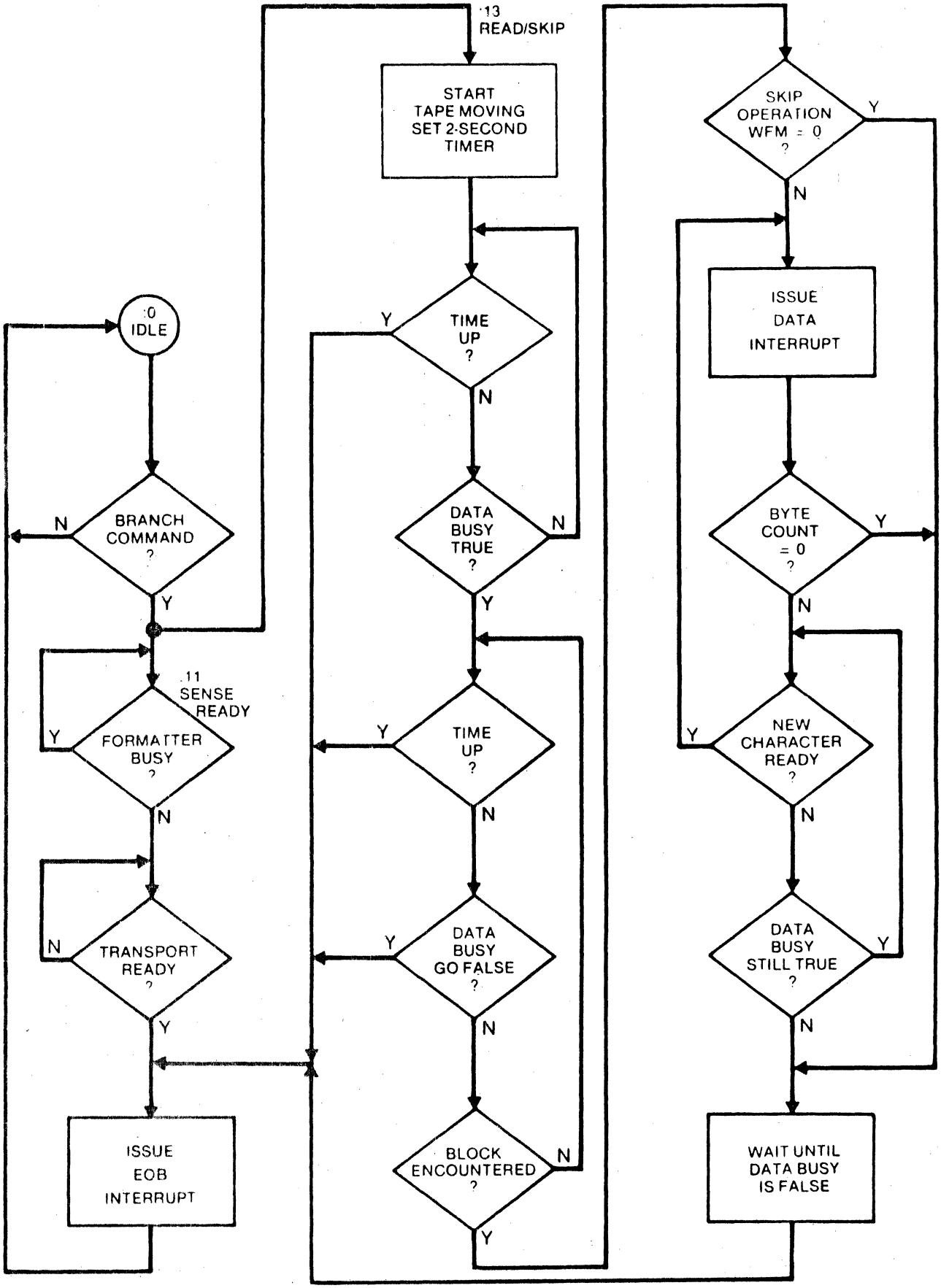



Figure 3-65b Read Firmware Sequence

# Table 3-10 Interface Cable Signal Assignments

| PADDLE BOARD CONNECTOR PIN NUMBER | PERTEC PIN NUMBER | DESCRIPTION | OTHER | PICO-PROCESSOR CONNECTION PAD NUMBER |
|---|---|---|---|---|
| 1 | B1 | Formatter Address (FAD) | Paddle Board Jumper | |
| 2 | A1 | Transport Address (TAD0) | | 46 |
| 3 | B2 | | | |
| 4 | A2 | Signal Ground (SIG GND) | | 23 |
| 5 | B3 | Transport Address (TAD1) | | 37 |
| 6 | A3 | Initiate Command (GO) | | 24 |
| 7 | B4 | Reverse/Forward (REV) | | 35 |
| 8 | A4 | Write/Read (WRT) | | 36 |
| 9 | B5 | | | |
| 10 | A5 | | | |
| 11 | B6 | Write File Mark (WFM) | | 38 |
| 12 | A6 | EDIT | | 39 |
| 13 | B7 | ERASE | | 40 |
| 14 | A7 | Read Threshold Level 1 (THR1) | Paddle Board Jumper | |
| 15 | B8 | | | |
| 16 | A8 | | | |
| 17 | B9 | Read Threshold Level 2 (THR2) | Paddle Board Jumper | |
| 18 | A9 | Density Select (DEN) | Paddle Board Jumper | |
| 19 | B10 | Parity Select (PAR | Paddle Board Jumper | |
| 20 | A10 | Load & On Line (LOL) | 1K Pull Up to + 5V | |
| 21 | B11 | Signal Ground (SIG GND) | | 29 |
| 22 | A11 | | | |
| 23 | B12 | Rewind (REW) | | 28 |
| 24 | A12 | Off-Line (OFL) | | 41 |
| 25 | B13 | Last Word (LWD) | | 34 |
| 26 | A13 | Formatter Enable (FEN) | | 31 |
| 27 | B14 | | | |
| 28 | A14 | | | |
| 29 | B15 | | | |
| 30 | A15 | | | |
| 31 | B16 | Write Data 0 (W0 | | 12 |
| 32 | A16 | Write Data 1 (W1) | | 5 |
| 33 | B17 | | | |
| 34 | A17 | | | |
| 35 | B18 | Write Data 2 (W2) | | 6 |
| 36 | A18 | Write Data 3 (W3) | | 7 |
| 37 | B19 | Write Data 4 (W4) | | 11 |
| 38 | A19 | Write Data 5 (W5) | | 10 |
| 39 | B20 | | | |
| 40 | A20 | | | |
| 41 | B21 | Write Data 6 (W6) | | 9 |
| 42 | A21 | Write Data 7 (W7) | | 16 |
| 43 | B22 | Formatter Bus, 8 BY | | 4 |
| 44 | A22 | Data Bus, DBY | | 3 |
| 45 | B23 | | | |
| 46 | A23 | | | |
| 47 | B24 | Identification Check Character Gate (IDENT CCG) | | 32 |
| 48 | A24 | Hard Error (HER | | 26 |
| 49 | B25 | Corrected Error (CER) | | 19 |
| 50 | A25 | File Mark FM+ | | 30 |

| PADDLE BOARD CONNECTOR PIN NUMBER | PERTEC PIN NUMBER | DESCRIPTION | OTHER | PICO-PROCESSOR CONNECTION PAD NUMBER |
|---|---|---|---|---|
| 51 | B26 | | | |
| 52 | A26 | | | |
| 53 | B27 | Ready (RDY) | | 1 |
| 54 | A27 | On Line (ONL) | | 14 |
| 55 | B28 | Rewinding (RWD) | | 2 |
| 56 | A28 | File Protect (FPT) | | 20 |
| 57 | B29 | | | |
| 58 | A29 | | | |
| 59 | B30 | Load Point (LDP) | | 8 |
| 60 | A30 | End Of Tape (EOT) | | 18 |
| 61 | B31 | | | |
| 62 | A31 | NRZI (NRZ) | | 33 |
| 63 | B32 | | | |
| 64 | A32 | | | |
| 65 | B33 | | | |
| 66 | A33 | | | |
| 67 | B34 | | | |
| 68 | A34 | Write Strobe (WSTR) | | 27 |
| 69 | B35 | Signal Ground (SIG GND) | | 25 |
| 70 | A35 | Signal Ground (SIG GND) | | 24 |
| 71 | B36 | Read Strobe (RSTR) | | 22 |
| 72 | A36 | | | |
| 73 | B37 | Read Data 0 (R0) | | 45 |
| 74 | A37 | Read Data 1 (R1) | | 44 |
| 75 | B38 | | | |
| 76 | A38 | | | |
| 77 | B39 | Read Data 2 (R2) | | 43 |
| 78 | A39 | Read Data 3 (R3) | | 42 |
| 79 | B40 | Read Data 4 (R4) | | 47 |
| 80 | A40 | Read Data 5 (R5) | | 48 |
| 81 | B41 | | | |
| 82 | A41 | | | |
| 83 | B42 | Read Data 6 (R6) | | 49 |
| 84 | A42 | Read Data 7 (R7) | | 21 |
| 85 | B43 | | | |
| 86 | A43 | | | |
| 87 | B44 | | | |
| 88 | A44 | | | |
| 89 | B45 | | | |
| 90 | A45 | | | |
| 91 | B46 | | | |
| 92 | A46 | | | |
| 93 | B47 | | | |
| 94 | A47 | | | |
| 95 | B48 | Power Ground (PWR GND | | 13 |
| 96 | A48 | + 5V | | |
| 97 | B49 | Power Ground (PWR GND) | | 15 |
| 98 | A49 | + 5V | | |
| 99 | B50 | Power Ground (PWR GND | | 26 |
| 100 | A50 | + 5V | | |

# Table 3-10 Interface Cable Signal Assignments

| PADDLE BOARD CONNECTOR PIN NUMBER | PERTEC PIN NUMBER | DESCRIPTION | OTHER | PICO-PROCESSOR CONNECTION PAD NUMBER |
|---|---|---|---|---|
| 1 | B1 | Formatter Address (FAD | Paddle Board Jumper | |
| 2 | A1 | Transport Address (TAD0) | | 46 |
| 3 | B2 | | | |
| 4 | A2 | Signal Ground (SIG GND) | | 23 |
| 5 | B3 | Transport Address (TAD1) | | 37 |
| 6 | A3 | Initiate Command (GO) | | 24 |
| 7 | B4 | Reverse/Forward (REV) | | 35 |
| 8 | A4 | Write/Read (WRT) | | 36 |
| 9 | B5 | | | |
| 10 | A5 | | | |
| 11 | B6 | Write File Mark (WFM) | | 38 |
| 12 | A6 | EDIT | | 39 |
| 13 | B7 | ERASE | | 40 |
| 14 | A7 | Read Threshold Level 1 (THR1) | Paddle Board Jumper | |
| 15 | B8 | | | |
| 16 | A8 | | | |
| 17 | B9 | Read Threshold Level 2 (THR2) | Paddle Board Jumper | |
| 18 | A9 | Density Select (DEN) | Paddle Board Jumper | |
| 19 | B10 | Parity Select (PAR) | Paddle Board Jumper | |
| 20 | A10 | Load & On Line (LOL) | 1K Pull Up to +5V | |
| 21 | B11 | Signal Ground (SIG GND) | | 29 |
| 22 | A11 | | | |
| 23 | B12 | Rewind (REW) | | 28 |
| 24 | A12 | Off-Line (OFL) | | 41 |
| 25 | B13 | Last Word (LWD) | | 34 |
| 26 | A13 | Formatter Enable (FEN) | | 31 |
| 27 | B14 | | | |
| 28 | A14 | | | |
| 29 | B15 | | | |
| 30 | A15 | | | |
| 31 | B16 | Write Date 0 (W0) | | 12 |
| 32 | A16 | Write Data 1 (W1) | | 5 |
| 33 | B17 | | | |
| 34 | A17 | | | |
| 35 | B18 | Write Data 2 (W2) | | 6 |
| 36 | A18 | Write Data 3 (W3) | | 7 |
| 37 | B19 | Write Data 4 (W4) | | 11 |
| 38 | A19 | Write Data 5 (W5) | | 10 |
| 39 | B20 | | | |
| 40 | A20 | | | |
| 41 | B21 | Write Data 6 (W6) | | 9 |
| 42 | A21 | Write Data 7 (W7) | | 16 |
| 43 | B22 | Formatter Busy (FBY) | | 4 |
| 44 | A22 | Data Busy (DBY) | | 3 |
| 45 | B23 | | | |
| 46 | A23 | | | |
| 47 | B24 | Identification/Check Character Gate (IDENT/CCG) | | 32 |
| 48 | A24 | Hard Error (HER) | | 26 |
| 49 | B25 | Corrected Error (CER) | | 19 |
| 50 | A25 | File Mark (FMK) | | 30 |

| PADDLE BOARD CONNECTOR PIN NUMBER | PERTEC PIN NUMBER | DESCRIPTION | OTHER | PICO-PROCESSOR CONNECTION PAD NUMBER |
|---|---|---|---|---|
| 51 | B26 | | | |
| 52 | A26 | | | |
| 53 | B27 | Ready (RDY) | | 1 |
| 54 | A27 | On Line (ONL) | | 14 |
| 55 | B28 | Rewinding (RWD) | | 2 |
| 56 | A28 | File Protect (FPT) | | 20 |
| 57 | B29 | | | |
| 58 | A29 | | | |
| 59 | B30 | Load Point (LDP) | | 8 |
| 60 | A30 | End Of Tape (EOT) | | 18 |
| 61 | B31 | | | |
| 62 | A31 | NRZI (NRZ) | | 33 |
| 63 | B32 | | | |
| 64 | A32 | | | |
| 65 | B33 | | | |
| 66 | A33 | | | |
| 67 | B34 | | | |
| 68 | A34 | Write Strobe (WSTR) | | 27 |
| 69 | B35 | Signal Ground (SIG GND) | | 25 |
| 70 | A35 | Signal Ground (SIG GND) | | 21 |
| 71 | B36 | Read Strobe (RSTR) | | 22 |
| 72 | A36 | | | |
| 73 | B37 | Read Data 0 (R0) | | 45 |
| 74 | A37 | Read Data 1 (R1) | | 44 |
| 75 | B38 | | | |
| 76 | A38 | | | |
| 77 | B39 | Read Data 2 (R2) | | 43 |
| 78 | A39 | Read Data 3 (R3) | | 42 |
| 79 | B40 | Read Data 4 (R4) | | 47 |
| 80 | A40 | Read Data 5 (R5) | | 48 |
| 81 | B41 | | | |
| 82 | A41 | | | |
| 83 | B42 | Read Data 6 (R6) | | 49 |
| 84 | A42 | Read Data 7 (R7) | | 50 |
| 85 | B43 | | | |
| 86 | A43 | | | |
| 87 | B44 | | | |
| 88 | A44 | | | |
| 89 | B45 | | | |
| 90 | A45 | | | |
| 91 | B46 | | | |
| 92 | A46 | | | |
| 93 | B47 | | | |
| 94 | A47 | | | |
| 95 | B48 | Power Ground (PWR GND) | | 3 |
| 96 | A48 | +5V | | |
| 97 | B49 | Power Ground (PWR GND) | | 15 |
| 98 | A49 | +5V | | |
| 99 | B50 | Power Ground (PWR GND) | | 17 |
| 100 | A50 | +5V | | |

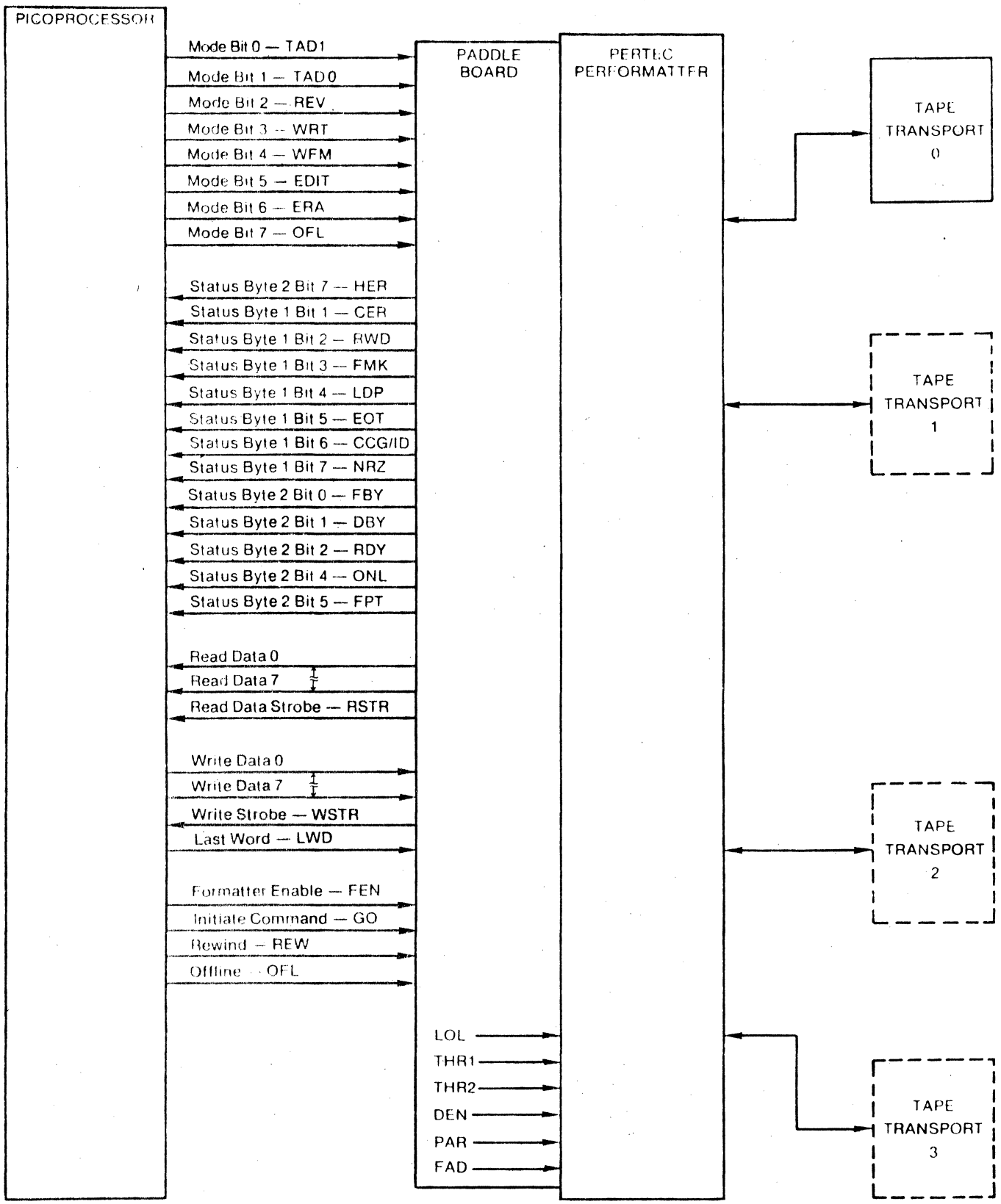
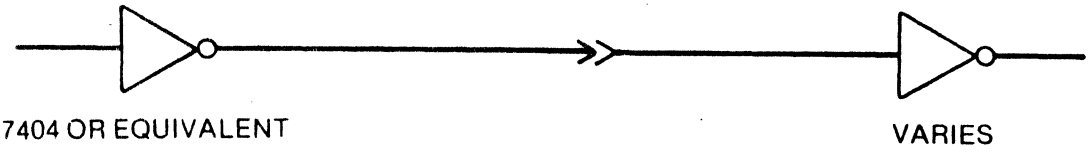


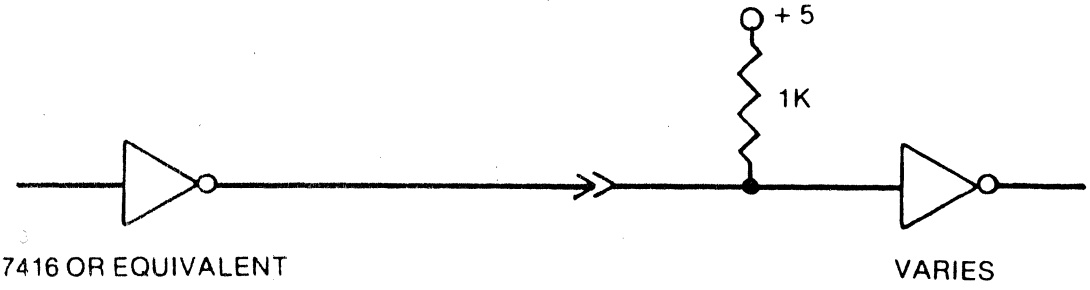

Figure 3-66 Interface Overview

The interface cable is a 50-conductor flat ribbon cable. The PICOPROCESSOR end is soldered directly to connector pads on the PICOPROCESSOR printed circuit board. A 100-pin female paddleboard connector is attached to the formatter end of the interface cable. Table 3-10 lists the signal assignments for the interface cable. Figure 3-66 presents an interface overview of the magnetic tape system.

Formatter outputs are driven by DTL 944, 932, or TTL 7416 open collector devices. The signals are pulled up in the PICOPROCESSOR to +5V through 1k OHM resistors and present an equivalent load of not more than one Schottky 74Sxx series load. Refer to Figure 3-67. Formatter inputs are DTL 936, 946 and TTL 74xx devices. The PICOPROCESSOR does not have power driver outputs and the formatter must not have terminating resistor voltage dividers on the input lines.




**TYPICAL PICOPROCESSOR OUTPUT**




**TYPICAL PICOPROCESSOR INPUT**

Figure 3-67 Typical Interface Lines

3.3.9.6 Programming Example

The MTIC programming example is included for illustrative purposes.

The Auto I/O instruction and the EOB Subroutine Locater are stored in the data service and EOB service areas as shown below.

```
*
*     SAMPLE AUTO I/O BLOCK
*     SET UP FOR READ MODE
*
MTINT     EQU       :C8          STANDARD MAG TAPE INTERRUPT
MTDEVA    EQU       :F2          STANDARD MAG TAPE DEVICE ADDRESS
          ABS       MTINT        INTERRUPT FOR DATA LOCATION
          AIB       MTDEVA       AUTO INPUT BYTE
          DATA      $-$          TO BE FILLED WITH BYTE COUNT
          DATA      $-$          TO BE FILLED WITH BUFFER ADDRESS -1
          DATA      0
          JST       *$+1         CALL EOB INTERRUPT HANDLER
          DATA      EOB          HANDLER ADDRESS
```

The Byte count and buffer address locations are filled in by the main section of code illustrated on the following page.

```
*          SAMPLE CODE FOR AUTO I/O READ OPERATION
*          WITH STANDARD DEVICE ADDRESS (:F) AND CHANNEL NUMBER (1)
*          AND TRANSPORT 0
*          HALTS ON ANY ERROR CONDITION
*
FBYLP      SIN       2
           INA       :F3          GET STATUS BYTE ONE
           INX       :F3          GET STATUS BYTE TWO
           AND       =:1          CHECK FOR PICO PRESENT (PRE=0)
           JAZ       $+2          PRESENT
           HLT                    NOT THERE-ERROR
           TXA
           LRA       1            CHECK FOR FORMATTER NOT BUSY
           JOR       FBYLP        NOT BUSY-GO ON WITH OPERATION
           LDA       :04FF        LOAD READ MODE WORD (INCLUDES TRANSPORT
                                  ADDRESS)
           OTA       :F3          MAKE SELECTION
RDYLP      SIN       2
           INX       :F3          GET STATUS BYTE ONE
           INA       :F3          GET STATUS BYTE TWO
           RRA       5            CHECK FOR TRANSPORT ON LINE
           JOR       $+2
           HLT                    NOT ON LINE-ERROR
           RLA       2            CHECK FOR TRANSPORT READY
           JOR       $+2
           JMP       RDYLP        NOT READY, LOOP
           LDA       XCNT         SET UP AUTO I/O

           STA       :C9          BYTE COUNT (NEGATIVE)
           LDA       XADDR        AND ADDRESS

           STA       :CA
           LDA       :230
           OTA       :F3
           JMP       $            WAIT FOR END-OF-BLOCK INTERRUPT
*
*          END-OF-BLOCK INTERRUPT HANDLER
*
EOB        ENT
           INX       :F3          GET THE STATUS
           INA       :F3
           XOR       =:C2         CHECK STATUS BYTE 2 FOR
           AND       =:D6         ONL AND RDY TRUE;HER,RAT,AND DBY FALSE
           JAZ       $+2
           HLT                    ERROR IN BYTE TWO
           END
```

### 3.3.9.7 Sample Magnetic Tape Bootstrap Program

A sample Magnetic Tape Bootstrap program using the standard device number, interrupt addresses, and standard channel number is shown below. The program reads the first Magnetic Tape record (regardless of size) into memory, beginning at location :200. The record is input from tape unit 0. Note that parameters DRVAD, DEVAD, CHNL, and INTAD can be changed to accomodate nonstandard addresses and other tape units.

The Bootstrap program is configured as a paper tape and can be loaded into memory using Autoload.

```
*
*          DIO MAGNETIC TAPE BOOTSTRAP
*
*
DRVAD     EQU       3              DRIVE ADDRESS 0 (NEGATIVE TRUE LOGIC)
DEVAD     EQU       :F             DEVICE ADDRESS
CHNL      EQU       1              CHANNEL #
INTAD     EQU       :C8            INTERRUPT ADDRESS
          ORG       INTAD-7
START     EQU       $
          LDA       MODSEL         PICK UP MODE SELECT WORD
          OTA       DEVAD*8+CHNL*2+1      SELECT MODE
          LDA       BRANCH         PICK UP BRANCH COMMAND WORD
          OTA       DEVAD*8+CHNL*2+1      START MTIC
          EIN                      ENABLE INTERRUPTS
          JMP       $              WAIT FOR EOB
          ORG       INTAD
          AIB       DEVAD*8+CHNL*2        AUTO INPUT INSTRUCTION
          DATA      0              READ COMPLETE RECORD REGARDLESS OF SIZE
          DATA      :3FF           STORE IT STARTING AT :200
          NOP
          JST       *$+1           GO TO START OF RECORD WHEN EOB RECEIVED
          DATA      :1FF           POINTER TO START OF
MODSEL    DATA      :4FC+DRYAD     MODE SELECT WORD - READ FORWARD
BRANCH    DATA      :0230          BRANCH COMMAND FOR MTIC TO START READ
END       START
```

### 3.3.10 IEEE Intelligent Cable

The IEEE Intelligent Cable (IEC), 14676-01, conforms to the requirements for an IEEE interface system controller. These requirements are contained in the IEEE document 488-1975, "IEEE Standard Digital Interface for Programmable Instrumentation". For convenience, this document is hereafter referred to as the IEEE Specification.

The IEC differs from most other Intelligent Cables in the nature of the PICOPROCESSOR to peripheral device interface. Most other Intelligent Cables interface a specific peripheral device or class of peripheral devices to the computer via the Distributed I/O System. The IEC interfaces the bus system defined by the IEEE Specification. Any device whose interface conforms to the IEEE Specification can be connected to the IEC. Up to 14 such devices can be connected to the bus controlled by the IEC.

The IEEE Specification uses specific definitions for the following words. To avoid confusion, these definitions are adhered to in this section.

Peripheral - A device that is IEEE 488-1975 Interface Bus compatible, and able to transmit and/or receive data over the Interface Bus. A Device with only periperal capabilities does not assume control of the Interface Bus.

Controller - A device that is programmed to have the responsibility of managing the flow of information on the IEEE 488-1975 Interface Bus. Since the Controller is required to both transmit and receive information, it can also participate in an information interchange as a Peripheral.

Talker - Any device that is addressed to place information on the data lines of the IEEE 488-1975 Interface Bus.

Listener - Any device that is addressed to accept information placed on the data lines by the Talker.

Local Message - Any information communicated between the device's internal system and it's interface section. In the Distributed I/O System, this is all communication between the Computer's I/O Distributor and the IEC PICOPROCESSOR.

Remote Message - Any information communicated between the interface section of one Peripheral (or Controller) and the interface section of another Peripheral.

3.3.10.1 IEEE Interface System Description

The IEEE Intelligent Cable (IEC) implementation of the IEEE Specification defines an interface bus system that consists of a system Controller (IEC) and at least one Peripheral. Up to 14 IEEE compatible Peripherals, in addition to the Controller (IEC), can be attached to a single IEEE 488-1975 Interface Bus. Other system peripherals that are not IEEE compatible can be interfaced to the computer through the I/O Distributor. See Figure 3-68.

The bus consists of 16 signal lines:

5 control lines
3 handshake lines
8 data lines

The bus interconnects the various Peripherals using a series of daisy-chain cables with stack connectors. The IEC connects to the bus at one of the stack connectors.

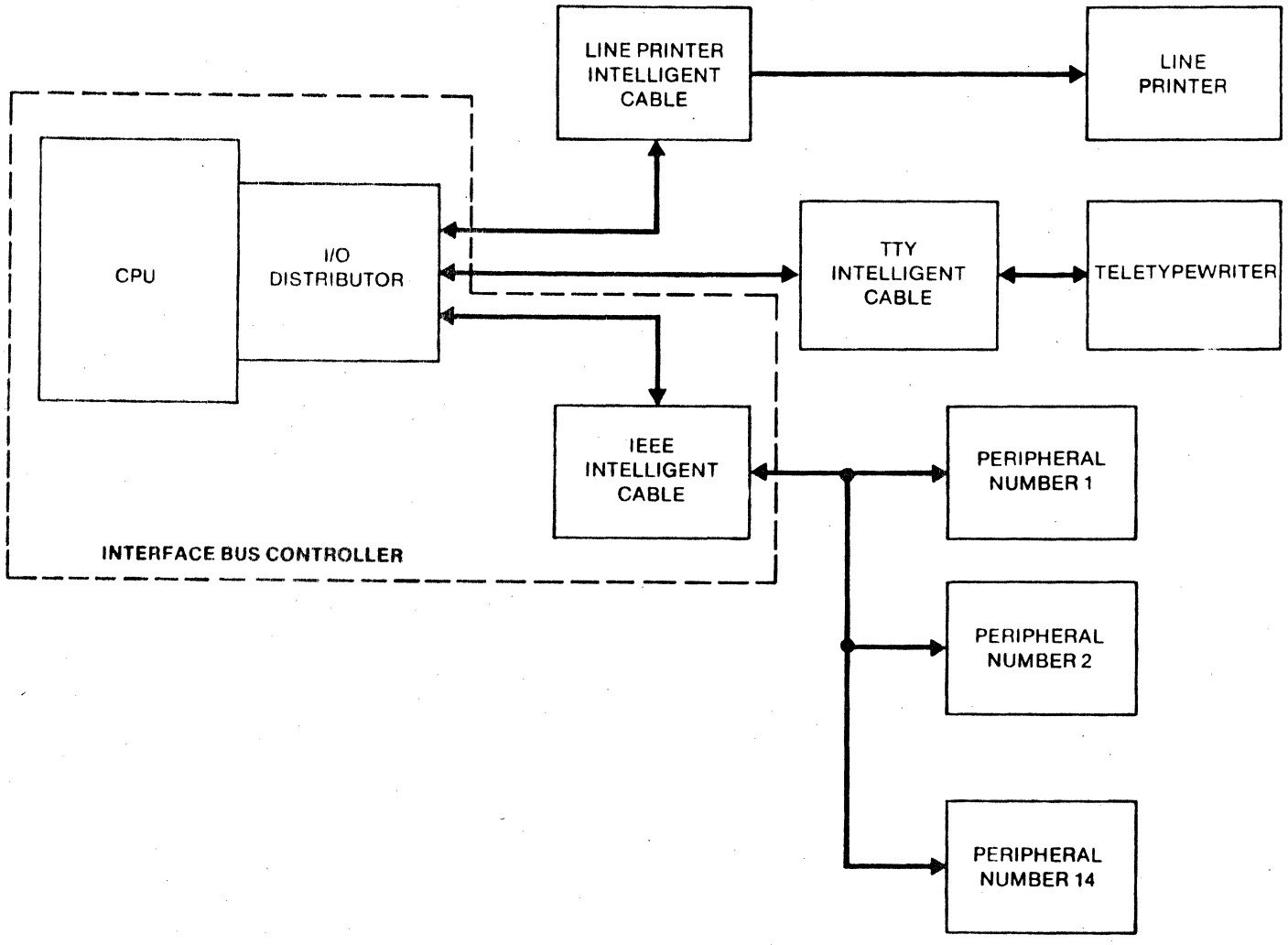



Figure 3-68. Typical System Configuration

Peripheral connected to the Interface Bus can have the capabilities to operate as both a Talker and a Listener or operation may be restricted to Listener only. The function of the Controller is to manage the flow of communication on the bus.

A Peripheral attached to the Interface Bus cannot be the Talker or a Listener unless it is instructed to be so by the controller. Each Peripheral has an individual Talker address and Listener address. These addresses are selectable and no two Peripherals can have the same address. Only one Peripheral may be addressed as the Talker; however, more than one peripheral may be addressed as simultaneous Listeners. The Controller may participate in a communication interchange as the Talker or as a Listener. The Controller may initiate the communication by designating a Talker and one or more Listeners, and then ignore the actual communication interchange.

A communication interchange consists of the transfer of one or more Remote Messages. Each Remote Message is coded by the state of one or more bus lines. Messages coded by a single bus line are called "uniline messages". Messages coded by more than one bus line are called "multiline messages". Two types of remote messages can be communicated over the IEEE Interface System:

Interface or command messages
Device dependent or data messages

Interface messages may be uniline or multiline, but device dependent messages must be multiline.

Interface messages affect the operation of the interface portion of a device. Most of the interface messages are sent by the Controller to the Peripherals. Examples of these messages can be expressed as "every Peripheral clear it's own interface portion" or "Peripheral with address A is the next Talker". A few interface messages are sent by the Peripheral to the Controller. Examples of these messages can be expressed as "Peripheral requests service by the Controller" or "end of message".

Device dependent messages are used to transfer measurement data and programming instructions. The programming instructions are usually one or more characters used by the Peripheral to select range settings, operation modes, etc. Measurement data is usually formatted as a real number including the appropriate sign, one or more digits and a floating or fixed decimal point. An exponent or overflow indicator may also be included.

3.3.10.1.1 IEC Functional Description

The IEC implements a subset of the IEEE Specification interface functions using a combination of hardware, PICOPROCESSOR firmware sequences, and computer software. This subset, in addition to the basic Peripheral functions, includes all of the Controller functions except those related to the passing of control to another Controller. The IEC must be the only Controller operating on the interface bus. If any of the Peripherals include Controller capabilities, these capabilities must be disabled and it's operation restricted to that of a Peripheral.

The IEC PICOPROCESSOR provides the hardware to drive the IEEE interface bus and the firmware to conduct both the Source Handshake and the Acceptor Handshake. The IEC PICOPROCESSOR also senses the state of the IEEE Interface Bus. This information is used in part to control the firmware sequences and, upon demand, is reported back to the computer as the channel status character.

The remaining functions that can be implemented through the IEC require the use of computer software designed for this purpose. With the computer software directly in control of the IEEE Interface Bus, any other Peripheral interfaced to the computer can be used in conjunction with the IEEE Interface Bus. For example, a teletypewriter attached to the computer through the Distributed I/O System can be used to request that certain tests be performed. A line printer can be used to print the results of the tests.

### 3.3.10.1.2 IEC Function Subset Description

The capabilities of a device that is IEEE 488-1975 compatible is described in terms of a subset of IEEE Specification functions implemented. The IEEE Specification describes, in detail, all the possible different functions. The following is a summary of the subset that describes the IEC. These descriptions include whether the function is implemented in hardware, software, or PICOPROCESSOR firmware.

SOURCE HANDSHAKE - SH1

The Source Handshake function is used to guarantee the proper transfer of multiline messages. The Source Handshake is combined with one or more Peripheral's Acceptor Handshake(s) and the resulting interlocked handshake sequence is used to asynchronously transfer each multiline message.

The Source Handshake in the IEC is controlled by a PICOPROCESSOR firmware sequence. The sequence is initiated by a Branch command word issued by the computer.

ACCEPTOR HANDSHAKE - AH1

The Acceptor Handshake function is used to guarantee the proper reception of a remote multiline message. The Acceptor Handshake may delay a multiline message transfer until it is ready to continue with the transfer process.

The Acceptor Handshake in the IEC is controlled by a PICOPROCESSOR firmware sequence. The sequence is initiated by a Branch command word issued by the computer.

TALKER - T4 and TE4

The Talker function is used to send device dependent messages over the interface to the Peripheral. There are two versions of the Talker function: The T function uses 1-byte addressing, whereas the TE function uses 2-byte addressing. Otherwise, both functions are the same.

The Talker function in the IEC is provided by the output of data from the computer with the IEC PICOPROCESSOR operating in the Source Handshake firmware sequence.

LISTENER - L2 and LE2

The Listener function is used to receive device dependent messages over the interface from a Peripheral. There are two versions of the Listener function: The L function uses 1-byte addressing, whereas the LE function uses 2-byte addressing. Otherwise, both functions are the same.

The Listener function in the IEC is provided by the input of data to the computer with the IEC PICOPROCESSOR operating in the Acceptor Handshake firmware sequence.

SERVICE REQUEST - SR0

The IEC does not generate a service request but has the capability to recognize a Service Request input from any Peripheral with SR1 capabilities. The Service Request is generated by a Peripheral to signal the IEC that service is required.

REMOTE LOCAL - RL0

The IEC has the capability to exercise remote programming control over any Peripheral that can accept remote programming (RL1 or RL2 capabilities). The IEC does not accept remote programming.

PARALLEL POLL - PP0

The IEC has the capability to configure any Peripheral with PP1 capabilities for a parallel poll, and the ability to then conduct a parallel poll of Peripheral with both PP1 and PP2 capabilities. The IEC cannot be configured for a parallel poll and does not respond to a parallel poll.

DEVICE CLEAR - DC0

The IEC has the capability to initialize any Peripheral attached to the Interface Bus that can accept device clear (DC1 or DC2 capabilities). The IEC does not accept initialization over the Interface Bus.

DEVICE TRIGGER - DT0

The IEC is able to initiate the operation of one or more Peripherals with DT1 capabilties by specific command (Group Execute Trigger, GET, remote message). The IEC does not accept the GET remote message.

CONTROLLER - C1, C2, C3, C4, and C25

The Controller functions are used to manage the message transfers on the Interface Bus. The following are the controller functions of the IEC.

System Controller - C1. This function specifies that the IEC has primary control responsibility for the Interface Bus. Since the IEC must be the only Controller on the bus, this function is inherent in the physical presence of the IEC on the bus.

Take Charge - C2. This function allows the IEC to clear and initialize the Interface Bus and take charge of the operation. This function is controlled by the computer software with the IEC PICOPROCESSOR firmware in source idle (:00).

Remote Program Enable - C3. This function allows the IEC to place Peripheral under remote programming control.

Respond To SRQ - C4. This function allows the IEC to respond when a Peripheral requests service from the Controller. This function is controlled by a PICO-PROCESSOR firmware sequence. The sequence is initiated by a Branch command word issued by the computer.

Controller - C25. This function subset includes sending interface messages conducting a parallel poll and taking control synchronously. This subset excludes all abilities to pass control to another Controller or to receive control from another Controller. The IEC must be the only Controller on the interface bus.

3.3.10.2 Specifications

Cable Length (Nominal): 16 Ft. (4.9 m)

I/O Distributor to PICOPROCESSOR, 4 Ft. (1.2 m)
PICOPROCESSOR to first Peripheral, 12 Ft. (3.7 m)

Data Types: 8-bit input and output

IEEE interface system function subsets:

| Implemented by IEC | | | | Peripheral Requirements |
|---|---|---|---|---|
| SH1 | L2 | PP0 | C2 | C0 |
| AH1 | LE2 | DC0 | C3 | All other functions, |
| T4 | SR0 | DT0 | C4 | no restrictions |
| TE4 | RL0 | C1 | C25 | |

Operating Mode, Half-duplex

Command Output, Two-command format with 6 mode bits

Status Input, 8 bit format

Standard I/O Distributor Channel Number, 3 (Device Address Field) :F6

Standard Data Service Vector Address, :D8

Standard End-of-Block Service Vector Address, :DC

Strapping Required, None

Power Requirements, +5V at 1.0 A

Environmental Requirements

Temperature, - 0° C to +50° C operating
-20° C to +100° C non-operating

Humidity, 5% to 95% relative, non-condensing

Maximum Transfer Rates (using Auto I/O programming)

LSI 2 Series

DMA I/O Distributor, 108 K bytes

Standard I/O Distributor, 64 K bytes

LSI 3/05

DMA I/O Distributor, 65 K bytes

Standard I/O Distributor 33 K bytes

### 3.3.10.3 Software Considerations

The IEC conforms to the general I/O Distributor programming philosophy described in Section 2. The IEC PICOPROCESSOR transfers data in 8-bit parallel bytes for both input and output.

The computer instructions for both data and control transfer are unchanged from standard usage. The two-word format used to transfer control information to the PICOPROCESSOR requires careful ordering of the command words output. In general, two command words are required to start each operation.

Computer Automation has available an RTX/IOX Software and Documentation Package (order number 19005-0X) to simplify programming for the IEC.

#### 3.3.10.3.1 IEC Command Word Set

The command word set used with the IEC consists of seven of the standard command words described in Section 2. The following is a list of the command words and applicable hexadecimal word skeletons used with the IEC:

Reset :0100
Branch :02-- (See Branch Address Field Description)
Set Mode :04-- (See Mode Field Description)
ASCII Carriage Return Detect :0A
Special Character Detect :2A
Load Special Character :80
Disable DMA :82- (See Branch Address Field Description)

The eight least significant bits (Bits 0 through 8) in the Branch and Set Mode command words are defined in this section for use with the IEC.

The Disable DMA command word is not required for use with the standard I/O Distributor and is treated as a Branch command word by the standard I/O Distributor.

The standard I/O Distributor does not provide for special character detection. The Load Special Character Command word is ignored by the standard I/O Distributor and the Special Character Detect Command word is treated as a branch command by the standard I/O Distributor. Special character detection is a required operation with the IEC, and must be provided by software when the IEC is used with the standard I/O Distributor.

#### 3.3.10.3.2 Mode Field Description

The IEC PICOPROCESSOR requires seven mode bits to control the IEEE Interface Bus. The Set Mode command word used with the IEC is the standard form that provides an 8-bit mode field. Figure 3-69 shows the configuration of the Set Mode command word. The functions controlled by each of the mode bits are as follows:

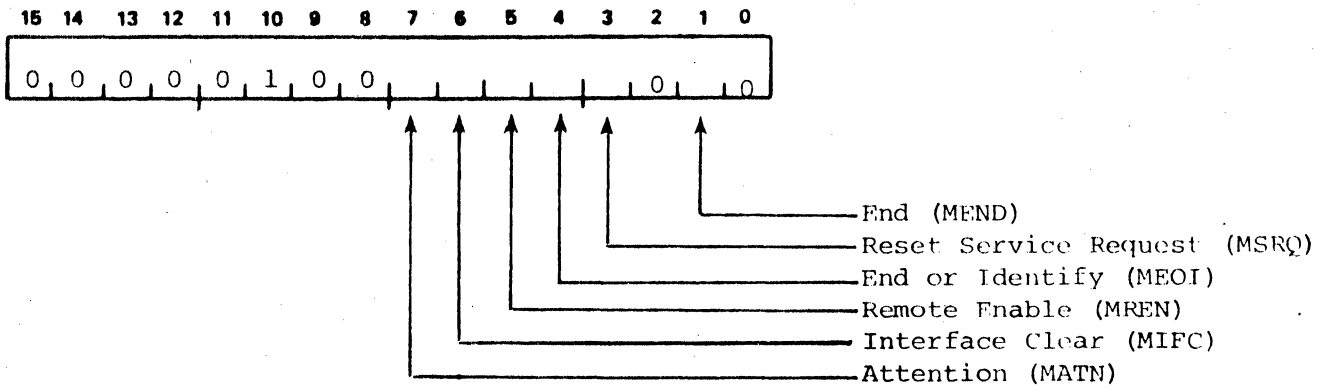



Figure 3-69. IEC Set Mode Command Word Format

Bit 1 - End (MEND). The MEND bit is used by the PICOPROCESSOR during the Source Handshake firmware sequence (Auto I/O programming). The logical 1 causes the PICOPROCESSOR to output the END remote message along with the last data byte. This signals the listener the last data byte is on the bus lines.

Bit 3 - Reset Service Request (MSRQ). The MSRQ bit is used to reset the PICOPROCESSOR firmware sequence if a service request is not received over the Interface Bus. The logical 1 causes the PICOPROCESSOR to generate a dummy service request to step the firmware to the end-of-block service request. The resulting interrupt to the computer software should be ignored.

Bit 4 - End or Identify (MEOI). The MEOI bit = logical 1 along with MATN = 1 is used to send the Identify (IDY) remote message over the Interface Bus. This message initiates a parallel poll. The MEOI bit = logical 1 when MATN = 0 is used to send the End remote message over the interface bus.

Bit 5 - Remote Enable (MREN). The MREN bit is used to send the Remote Enable (REN) remote message over the Interface Bus. The logical 1 signals an addressed Peripheral on the bus to accept programming (control settings, etc.) from the IEC rather than from their front panel controls.

Bit 6 - Interface Clear (MIFC). The MIFC bit is used to send the Interface Clear (IFC) remote message over the Interface Bus. The logical 1 signals all Peripherals on the bus to initialize their interface logic.

Bit 7 - Attention (MATN). The MATN bit = logical 1 is used to send the Attention (ATN) remote message over the Interface Bus. The MATN bit is normally set to the Logical 1 by a Set Mode command word. During a take control synchronously operation, the MATN bit is automatically set to a logical 1 by the PICOPROCESSOR.

3.3.10.3.3 Branch Address Field Description

The IEC PICOPROCESSOR requires a 5-bit branch address plus three additional flag bits to control the PICOPROCESSOR firmware and hardware. The branch command words used with the IEC are standard command words with Bits 0 through 7 defined specifically for the IEC. Figure 3-70 shows the configuration of these command words. The functions controlled by Bits 0 through 7 are as follows:

Bits 0 through 4 - Branch Address. The configuration of the five branch address bits is used to branch the PICOPROCESSOR firmware from idle (:00 or :10) to one of the allowed entry points to the firmware sequence.

Bit 5 - Take Control Synchronously (FTCS). The FTCS bit initiates the IEEE Interface Bus take control synchronously operation. The logical 1 causes the PICOPROCESSOR to sense for the time during a handshake cycle when all the Peripherals are unable to receive data. When this occurs, the MATN bit is set to logical 1 and the IEC assumes control of the Interface Bus.

NOTE

The firmware must be in idle (:00) before take control synchronously is issued and take control synchronously must be performed in idle :00.

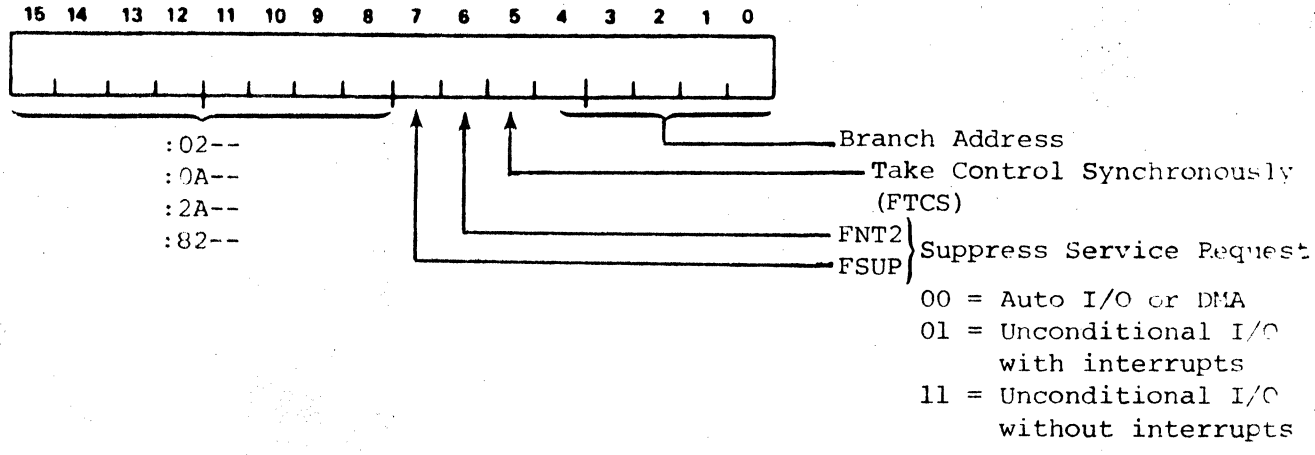



Figure 3-70. IEC Branch Command Word(s)

Bit 6 - Suppress Data Service Requests (FNT2). The logical 1 in the FNT2 bit location causes the PICOPROCESSOR to suppress the data service requests to the I/O Distributor. An End-of-Block service request is issued by the PICOPROCESSOR following the transfer of each data byte. FNT2 can be used with both the Source and Acceptor Handshake.

Bit 7 - Suppress End-of-Block Service Requests (FSUP). The logical 1 in the FSUP bit location causes the PICOPROCESSOR to suppress the End-of-Block service requests to the I/O Distributor. Following each data byte transfer, the PICOPROCESSOR firmware returns to idle (:00). FNT2 must be a logical 1 anytime FSUP is a logical 1.

### 3.3.10.3.4 Status Information

The IEEE Intelligent Cable's PICOPROCESSOR provides eight bits of status information for the computer. The IEC status bits are arranged in a single status byte. The individual status bits are derived from either the PICOPROCESSOR hardware or the Interface Bus control lines.

The bit assignments for the status byte are shown in Figure 3-71. The following describes the conditions reported by each bit.

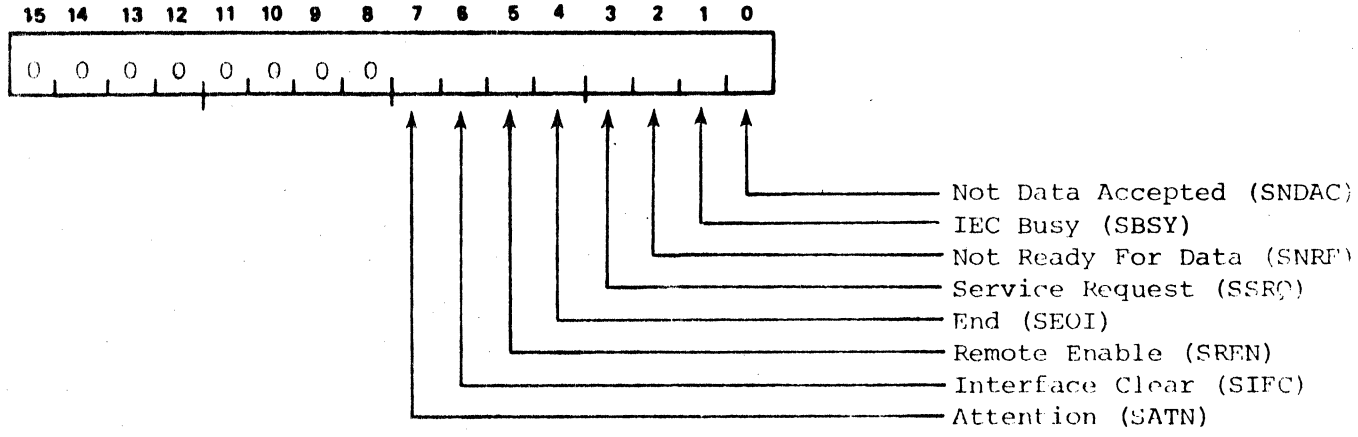



Figure 3-71. Status Byte Configuration

Bit 0 - Not Data Accepted (SNDAC). The SNDAC bit indicates the current value of Not Data Accepted (NDAC) on the Interface Bus. This bit is provided for diagnostic purposes only.

Bit 1 - IEC Busy (SBSY). The SBSY bit is a logical 1 anytime the IEC PICO-PROCESSOR is executing a firmware sequence. A command word should not be issued by the computer unless SBSY is a logical 0.

Bit 2 - Not Ready For Data (SNRF). The SNRF bit indicates the currect value of Not Ready For Data (NRFD) on the Interface Bus. This bit is provided for diagnostic purposes only.

Bit 3 - Service Request (SSRQ). The SSRQ bit indicates the currect value of Service Request (SRQ) on the Interface Bus. This bit is a logical 1 whenever a Peripheral is requesting service. The line does not reset to a logical 0 until the Controller (IEC) conducts a serial poll and services the request.

Bit 4 - End Or Identify Indicator (SEOI). The SEOI bit is a logical 1 if an END message was received over the Interface Bus during the preceding Acceptor Handshake cycle. This indicates the END message was transferred along with the most recent data byte.

Bit 5 - Remote Enable (SREN). The SREN bit indicates the currect value of the Remote Enable (REN) on the Interface Bus. This bit is provided for diagnostic purposes only.

Bit 6 - Interface Clear (SIFC). The SIFC bit indicates the current value of Interface Clear (IFC) on the Interface Bus. This bit is provided for diagnostic purposes only.

Bit 7 - Attention (SATN). The SATN bit indicates the current value of Attention (ATN) on the Interface Bus. This bit is provided for diagnostic purposes only.

### 3.3.10.4 Operating Sequence

The operation of the IEEE Interface is controlled by the computer software together with operation sequencing provided by the PICOPROCESSOR firmware. Selection of the appropriate PICOPROCESSOR firmware sequence is made by the computer software.

#### 3.3.10.4.1 Remote Messages

The IEEE Specification defines the Interface Bus operation in terms of remote messages sent and received. The IEEE Specification defines a list of remote messages and requires that no new remote messages be defined. Operation using only a subset of the remote messages is allowed.

The IEEE Specification divides the remote messages into uniline messages and multiline messages. Within the protocol of the Interface Bus, combining of messages--both uniline and multiline--is allowed. In many cases, the combining of messages is mandatory.

The IEEE specification provides a table, similar to Table 3-11, listing all the remote messages and the specific coding for each message. Table 3-11 interprets the IEEE Specification remote message coding table into the local message coding used with the IEC. To aid in relating the listed local message to the corresponding Interface Bus remote message, an "L" (for local) is added to the corresponding remote message mnemonic. The data bits 0 through 7 correspond to the Interface Bus lines DIO1 through DIO8 respectively. The Mode Bits 4 through 7 correspond to the Interface Bus lines EOI, REN, IFC, and ATN. The Interface Bus lines DAV, NRFD, and NDAC are controlled and sequenced by the PICOPROCESSOR Source (SH) and Acceptor (AH) Handshake firmware sequences. The notes in the IEEE Specification remote message coding table referring to the mandatory combining of uniline messages are incorporated in Table 3-11 as mandatory 1 or 0 bits.

#### 3.3.10.4.2 Remote Message Sequences

The Interface Bus protocol established in the IEEE Specification must be followed when preparing programs for use with the IEC. This section describes a few of the most common remote message sequences. The sequences shown here concentrate on the programming for the IEEE remote messages, and not on the Distributed I/O programming.

In these descriptions, the IEC is assumed in idle with the IEC in control of the Interface Bus at the start of each sequence. The IEC is in control of the Interface Bus anytime MATN = 1.

INITIALIZATION

Initialization is used to prepare the Peripherals on the Interface Bus for subsequent operation. Full initialization requires two separate remote messages be transferred. Figure 3-72 shows the flow chart for initialization.

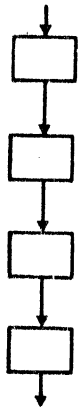
# Table 3-11. Message Coding

| MNEMONIC | MESSAGE NAME | DATA BITS 7 6 5 4 3 2 1 0 | MODE BITS 7 6 5 4 (ATN IFC REN EOI) | FIRMWARE SEQUENCE | REMARKS |
|---|---|---|---|---|---|
| LACG | Addressed Command Group | X 0 0 0 X X X X | 1 0 X 0 | SH | |
| LATN | Attention | X X X X X X X X | 1 0 X 0 | None Required | |
| LDAB | Data Byte | | X 0 X X | SH,AH | Data bits as required |
| LDAC | Data Accepted | X X X X X X X X | X 0 X X | AH | |
| LDAV | Data Valid | X X X X X X X X | X 0 X X | SH | |
| LDCL | Device Clear | X 0 0 1 0 1 0 0 | 1 0 X 0 | SH | |
| LEND | End | X X X X X X X X | 0 0 X 1 | SH,AH | |
| LEOS | End of String | | 0 0 X X | SH,AH | Data bits = Special End Character |
| LGET | Group Execute Trigger | X 0 0 0 1 0 0 0 | 1 X X 0 | SH | |
| LGTL | Go to Local | X 0 0 0 0 0 0 1 | 1 0 0 0 | SH | |
| LIDY | Identify | X X X X X X X X | 1 0 X 1 | None Required | |
| LIFC | Interface Clear | X X X X X X X X | 0 1 X X | None Required | |
| LLAG | Listen Address Group | X 0 1 X X X X X | 1 0 X 0 | SH | |
| LLLO | Local lock Out | X 0 0 1 0 0 0 1 | 1 0 0 0 | SH | |
| LMLA | My Listen Address | X 0 1 | 1 0 X 0 | SH | Data bits 0 - 4 = Address |
| LMTA | My Talk Address | X 1 0 | 1 0 X 0 | SH | Data bits 0 - 4 = Address |
| LMSA | My Secondary Address | X 1 1 | 1 0 X 0 | SH | Data bits 0 - 4 = Address |
| LNUL | Null Byte | 0 0 0 0 0 0 0 0 | 0 0 X 0 | SH,AH | |
| LPPC | Parallel Poll Configure | X 0 0 0 0 1 0 1 | 1 0 X 0 | SH | |
| LPPE | Parallel Poll Enable | X 1 1 0 | 1 0 X 0 | SH | Bit 3 = Sense, Bits 0 - 2 = PPR(N) |
| LPPD | Parallel Poll Disable | X 1 1 1 0 0 0 0 | 1 0 X 0 | SH | |
| LPPR1 | Parallel Poll Response 1 | X X X X X X X 1 | 0 0 X 0 | AH | |
| LPPR2 | Parallel Poll Response 2 | X X X X X X 1 X | 0 0 X 0 | AH | |
| LPPR3 | Parallel Poll Response 3 | X X X X X 1 X X | 0 0 X 0 | AH | |
| LPPR4 | Parallel Poll Response 4 | X X X X 1 X X X | 0 0 X 0 | AH | |
| LPPR5 | Parallel Poll Response 5 | X X X 1 X X X X | 0 0 X 0 | AH | |
| LPPR6 | Parallel Poll Response 6 | X X 1 X X X X X | 0 0 X 0 | AH | |
| LPPR7 | Parallel Poll Response 7 | X 1 X X X X X X | 0 0 X 0 | AH | |
| LPPR8 | Parallel Poll Response 8 | 1 X X X X X X X | 0 0 X 0 | AH | |
| LPPU | Parallel Poll Unconfigure | X 0 0 1 0 1 0 1 | 1 0 X 0 | SH | |
| LREN | Remote Enable | X X X X X X X X | X 0 1 0 | None Required | |
| LRFD | Ready for Data | X X X X X X X X | X 0 X X | SH,AH | |
| LRQS | Request Service | X 1 X X X X X X | 0 0 X 0 | AH | |
| LSCG | Secondary Command Group | X 1 1 X X X X X | 1 0 X 0 | SH | |
| LSDC | Selected Device Clear | X 0 0 0 0 1 0 0 | 1 0 X 0 | SH | |
| LSPD | Serial Poll Disable | X 0 0 1 1 0 0 1 | 1 0 X 0 | SH | |
| LSPE | Serial Poll Enable | X 0 0 1 1 0 0 0 | 1 0 X 0 | SH | |
| LSRQ | Service Request | X X X X X X X X | 0 0 X 0 | SRQ | Input service requests only |
| LSTB | Status Byte | | 0 0 X 0 | AH | Bits 1 - 5 and bit 7 = status, bit 6 = RQS |
| LTAG | Talk Address Group | X 1 0 X X X X X | 1 0 X 0 | SH | |
| LUCG | Universal Command Group | X 0 0 1 X X X X | 1 0 X 0 | SH | |
| LUNL | Unlisten | X 0 1 1 1 1 1 1 | 1 0 X 0 | SH | |
| LUNT | Untalk | X 1 0 1 1 1 1 1 | 1 0 X 0 | SH | |

X = don't care. These values may be ignored or treated as a combining of remote messages. For example, PPR1 through PPR8 are always input as a single remote message.

Mode bits listed are mode register contents for that specific remote message.

Data bits are output required with source handshake (SH) and input expected with acceptor handshake (AH).



Issue the Set Mode command word for the Interface Clear (IFC) remote message.

Output the data portion of the Device Clear (DCL) remote message.

Issue the Set Mode command for the DCL remote message.

Issue a Branch command word to the Source Handshake firmware sequence (Unconditional I/O).

Figure 3-72. Initialization Flow Chart

The first remote message, Interface Clear (IFC), requires only the setting of the mode register. The IFC message clears the interface portion of all Peripherals on the Interface Bus, leaving them ready to accept further messages from the Controller (IEC).

The second remote message, Device Clear (DCL), is usually sent immediately after the IFC message. The DCL message requires the setting of the mode register and branching to the Source Handshake firmware sequence. The flow chart in Figure 3-72 assumes the data portion of the DCL message will be output using Unconditional I/O. For Unconditional I/O, the data must be output and stored in the PICOPROCESSOR before the Branch command word is issued.

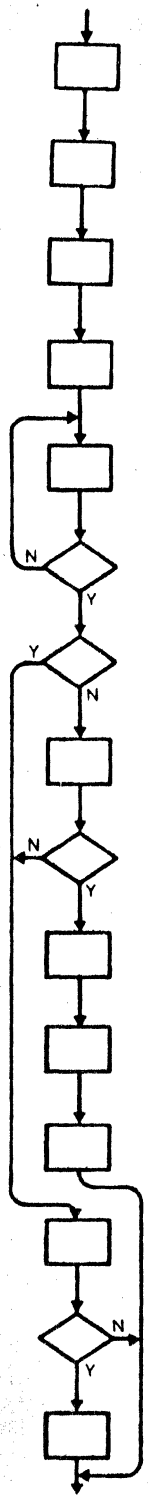
ADDRESSING

Each Peripheral on the Interface Bus has a unique address. When the controller wants a Peripheral to talk, it sends the remote message My Talk Address (MTA) with the Peripheral's device address in data bits 0 through 4. Only that Peripheral whose address appears in bits 0 through 4 will accept the MTA message.

Similarly, when the Controller wants a Peripheral to listen, it sends the remote message My Listen Address (MLA) with the Peripheral's device address in data bits 0 through 4. Only that Peripheral whose address appears in bits 0 through 4 will accept the MLA message.

The Controller can address more than one Peripheral as a Listener but only one Peripheral can be addressed as the Talker. The Controller can be a Listener, the Talker, or neither. Figure 3-73 is the addressing flow chart.

Unlisten (UNL), Untalk (UNT), My Listen Address (MLA), and My Talk Address (MTA) all use the same mode register configuration. The flow chart in Figure 3-73 shows a Set Mode command word being issued. If MATN is changed from 0 to 1 by this command word, the result would be the remote message Take Control Asynchronously. Since this is not recommended, a Take Control Synchronously message may be needed before starting this flow.




Issue the set mode command word for the Unlisten (UNL), Untalk (NNT), My Listen Address (MLA) and My Talk Address (MTA) remote messages.

Issue a Branch command word to the Source Handshake firmware sequence (Auto I/O or DMA).

Output the data portion of the UNL remote message at the first data service request.

Output the data portion of the UNT remote message at the second data service request.

Output the data portion of an MLA remote message at the next data service request.

Are all listeners addressed?

Is the IEC the talker?

Output the data portion of an MTA remote message at the last data service request.

Is the IEC a listener?

Issue a Branch command word to Acceptor Idle :10.

Issue a Set Mode command word with MATN = 0.

Issue a Branch command word to the Acceptor Handshake firmware sequence.

Issue a Set Mode command word with MATN = 0.

Is the IEC the talker?

Issue a branch command word to the Source Handshake firmware sequence.

Figure 3-73. Addressing Flow Chart

The data portion of the remote messages can then be output using a single Auto I/O instruction. When all the addressing is completed, a Set Mode command word must be issued to change MATN from 1 to 0. This terminates the addressing sequence and allows data transfer to begin.

If the IEC is to act as a listener or the talker, a series of local messages must be transferred between the computer and the IEC before MATN = 0 is issued. These local messages are required to initiate the appropriate handshake firmware sequence. For the Acceptor Handshake, a special Branch command word from source idle :00 to acceptor idle :10 must be issued. The acceptor idle also inhibits data transfer on the Interface Bus. The Set Mode command word can now be issued, followed by the Branch command word to the acceptor handshake firmware sequence without a risk of losing data.

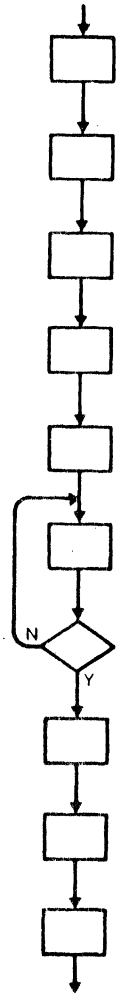
REMOTE CONTROL

Some IEEE compatible Peripherals can be programmed remotely via the Interface Bus. To operate a Peripheral by remote control, the Controller must continuously send the remote message Remote Enable (REN). Anytime the Controller stops sending the REN remote message, all Peripherals on the Interface Bus revert to local (front panel) control. Figure 3-74 is the flow chart for a remote programming sequence.

The first part of the flow chart is a simplified addressing sequence based on the assumption that a single Peripheral is addressed as listener and the Controller (IEC) is the talker. Once the Peripheral is addressed as listener, a Set Mode command word is issued to change MATN from 1 to 0. The computer then issues a Branch command word to the Source Handshake firmware sequence and transfers the necessary programming information.

The meaning of the programming information, the number of bytes required, and the means of distinguishing programming information from data is dependent on the specific Peripheral device. Refer to the applicable Peripheral user's manual.

Once all the programming information is transferred, the computer sends the Unlisten (UNL) remote message to terminate the operation with the addressed Peripheral. This frees the Interface Bus for use with another Peripheral.




Issue the Set Mode command word for the My Listen Address (MLA) remote message with MREN = 1.

Output the data portion of the MLA remote message.

Issue a Branch command word to the Source Handshake firmware sequence (Unconditional I/O).

Issue a Set Mode command word for transfer of the programming bytes (MATN = 0, MREN = 1).

Issue a Branch command word to the Source Handshake firmware sequence (Auto I/O).

Output a byte of programming information.

Are all programming bytes transferred?

Output the data portion of the UNL remote message.

Issue the Set Mode command word for the Unlisten (UNL) remote message with MREN = 1.

Issue a Branch command word to the Source Handshake firmware sequence (Unconditional I/O).

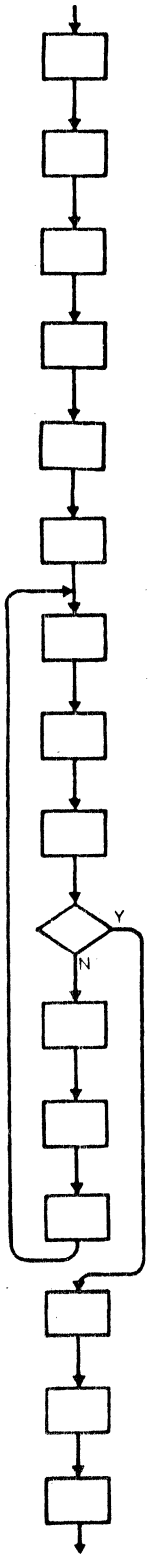
Figure 3-74. Remote Control Programming Flow Chart

SERVICE REQUEST AND SERIAL POLL

After remote control programming or when otherwise instructed by the Controller (IEC), some IEEE compatible Peripherals can perform a task such as taking a measurement off-line. If this Peripheral is not addressed as the Talker when the data is ready, it is capable of sending a Service Request (SRQ) remote message to the Controller. When the Controller recognizes that a service request is pending (see firmware sequence, wait for service request), it conducts a serial poll to determine which Peripheral is requesting service. Figure 3-75 is the flow chart for a serial poll sequence.

The Controller (IEC) prepares for the serial poll by outputting this sequence of remote messages: Unlisten (UNL), Untalk (UNT) Serial Poll Enable (SPE), and My Talk Address (MTA). These three remote messages can use the same mode register setting and can be output with a single Auto I/O instruction. The UNL and UNT messages prepare all Peripherals for control input. The SPE message signals all Peripherals that status input is required. The MTA message addresses one of the Peripherals as Talker. Since this can be any of the Peripherals on the Interface Bus, the serial poll can be conducted in a priority order or, to conserve time, the Peripheral most likely to be requesting service can be polled first.

Once the Talker is addressed, the Controller prepares to listen to the addressed Talker. When the status byte is received, Data Bit 6 is examined for the Request Service (RQS) remote message. The remaining bits in the status byte are dependent on the specific Peripheral involved. If Bit 6 = 0, the Talker Peripheral is not requesting service. The Controller then addresses another Peripheral as Talker and continues the serial poll. If Bit 6 = 1, the talker Peripheral is requesting service. The Controller then terminates the serial poll by sending the Serial Poll Disable (SPD) remote message and services the detected service request.




Issue the Set Mode command word for the Unlisten (UNL), Untalk (UNT) Serial Poll Enable (SPE), and My Talk Address (MTA) remote messages.

Issue a branch command word to the Source Handshake firmware sequence (Auto I/O or DMA).

Output the data portion of the UNL remote message at the first data service request.

Output the data portion of the UNT remote message at the second data service request.

Output the data portion of the SPE remote message at the third data service request.

Output the data portion of the MTA remote message at the fourth data service request.

Issue a Branch command word to Acceptor Idle :10.

Issue a Set Mode command word with MATN = 0.

Issue a Branch command word to the Acceptor Handshake firmware sequence (Unconditional I/O).

Did the Peripheral send an RQS remote message (bit 6 = 1)?

Output the data portion of a MTA remote message.

Issue the Set Mode command word for the MTA remote message.

Issue a Branch command word to the Source Handshake firmware sequence (Unconditional I/O).

Output the data portion of the Serial Poll Disable (SPD) remote message.

Issue the Set Mode command word for the SPD remote message.

Issue a Branch command word to the source handshake firmware sequence (Unconditional I/O).

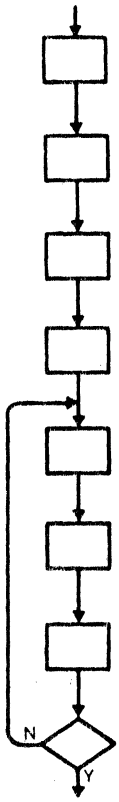
Figure 3-75. Serial Poll Flow Chart

PARALLEL POLL

The alternative to serial polling is parallel polling. Peripherals that are able to respond to a parallel poll input a single status bit during the parallel poll. The meaning of this one status bit is determined by the Peripheral. Any Peripheral that can respond to the parallel poll must be able to transmit its status bit as any one of the eight data bit and as either a positive true (equivalent to ANDing) or negative true (equivalent to ORing) signal. More than one Peripheral can be assigned to the same response data line. Figure 3-76 is the flow chart for the parallel poll configuration operation. The actual parallel poll is conducted using a single remote message at a later time when the status information is required.

The entire parallel poll configuration operation uses the same setting of the mode register and all the remote messages can be transferred using a single Auto I/O instruction. The UNL remote message prepares all Peripherals for control input. The Parallel Poll Configure (PPC) remote message prepares all Peripherals for a parallel poll configuration operation. The MLA remote message selects a specific Peripheral for parallel poll configuration. The Peripheral can then be disabled from responding to a parallel poll by the Parallel Poll Disable (PPD) remote message, or can be enabled to respond to a parallel poll by the Parallel Poll Enable (PPE) remote message. In the PPE remote message, Data Bits 0 through 2 provide the binary form of the data bit number used by the Peripheral during a parallel poll. Data Bit 3 specifies the response sense to be used, negative true = 0 and positive true = 1.

The Controller need only configure the Peripheral from which a response is required. A reconfiguration can involve just one Peripheral, or all of the Peripherals.




Issue a Set Mode command word for the parallel poll configuration operation.

Issue a Branch command word to the Source Handshake firmware sequence (Auto I/O or DMA).

Output the data portion of the Unlisten (UNL) remote message at the first data service request.

Output the data portion of the Parallel Poll Configure (PPC) remote message at the next service request.

Output the data portion of a My Listen Address (MLA) remote message at the next data service request.

Output the data portion of a Parallel Poll Enable (PPE) or Disable (PPD) remote message at the next data service request.

Output the data portion of the Unlisten (UNL) remote message at the next data service request.

Is the parallel poll configuration operation completed?

Figure 3-76. Parallel Poll Configure

### 3.3.10.4.3 PICOPROCESSOR Firmware Sequences

Once initiated, the PICOPROCESSOR firmware sequences assume control of the I/O operation until terminating conditions are sensed. The IEC firmware consists of two main sequences: the Source Handshake and the Acceptor Handshake plus a Wait-for-Service request sequence. The following describes all the various firmware sequences. Figure 3-77a and 3-77b are the flow charts for the firmware sequences.

#### IDLE (:00 or :10)

The PICOPROCESSOR firmware loops in Idle whenever a firmware sequence is not being performed. The firmware must be in Idle to accept a Branch command word to one of the sequences.

Idle :00 is called the Source Idle. Initializing the PICOPROCESSOR logic (Reset command word or MRST = 1) resets the firmware to Idle :00. The firmware also returns to Idle :00 following a Source Handshake or a Wait-for-Service request firmware sequence. The Take Control Synchronously operation takes place with the firmware in Idle :00. The computer can issue a Branch Command word to change the firmware from Idle :10 to Idle :00.

Idle :10 is called the Acceptor Idle. The firmware returns to Idle :10 following the Acceptor Handshake firmware sequence. The computer can also issue a branch command word to change the firmware from Idle :00 to Idle :10.

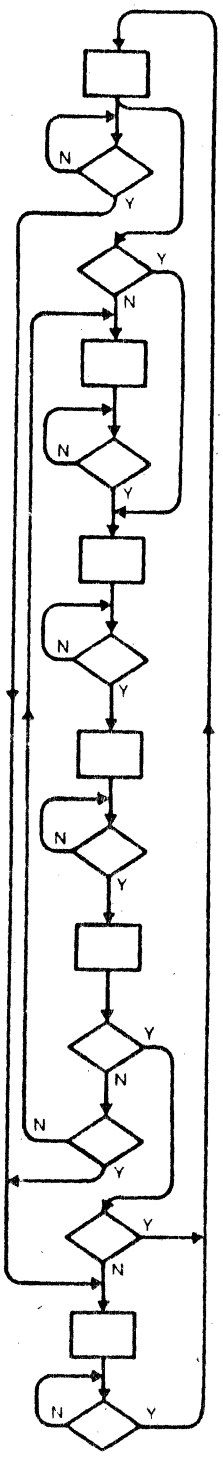
The PICOPROCESSOR operation is the same in either idle state; however, Interface Bus control is different. In the Source Idle :00, data transfers on the Interface Bus can take place uninhibited. In the Acceptor Idle :10, data transfers on the Interface Bus are inhibited. The busy status bit normally = 0 when the firmware is in Idle; However, for 2 $\mu$s following a successful Take Control Synchronously operation, Busy = 1.

#### SOURCE HANDSHAKE (:01)

The Source Handshake firmware sequence is used any time information is output on the Interface Bus data lines. The information output may be data, control information, or programming. When the PICOPROCESSOR is in the source handshake firmware sequence, the IEC is the Interface Bus talker.

The Source Handshake firmware sequence first checks whether the operation is Unconditional I/O. If it is (FNT2 = 1), the data is assumed already in the PICOPROCESSOR data register. If not, a data character must be requested. When data is available, the contents of the data register is loaded onto the Interface Bus data lines and a 2 sec delay is initiated. Following the delay, the firmware sequence waits until all listeners signal ready for data (NRFD = 0). Data valid is output then and the firmware sequence again waits until the listeners signal the data is accepted (NDAC = 0). The firmware sequence then makes a series of checks:

> If the operation is Unconditional I/O (FNT2 = 1), the firmware checks for interrupts being suppressed (FSUP = 1). If they are, the firmware returns to Idle :00. If not, an End-Of-Block service request is issued to the I/O Distributor. When the End-Of-Block is acknowledged, the firmware returns to Idle :00.




Idle - firmware loops in :00 until a branch command is received to indicate a firmware sequence.

Entry point :0E
Is any device requesting service?

Entry point :01
Is this operation Unconditional I/O? (FNT2 = 1).

Issue a data service request to the I/O Distributor.

Was the request acknowledged?

Load the data output lines, wait for 2 $\mu$sec (IEEE line settling requirement).

Is the listener device ready? (NRFD = 0).

Send the data valid signal to the listener. (DAV = 1).

Was the data accepted? (NDAC = 0).

Remove the data valid signal to the listener. (DAV = 0)

Is this operation Unconditional I/O? (FNT2 = 1).

Is the byte count = 0?

Are the PICOPROCESSOR End-of-Block service requests suppressed? (FSUP = 1).

Issue an End-of-Block service request to the I/O Distributor.

Was the request acknowledged?

Figure 3-77a. Source Firmware Sequence Flow Chart

If the operation is Auto I/O or DMA, the byte count = 0 is checked. If it is not, the firmware sequence issues a data service request and continues transferring data until byte count = 0 is reached. An End-Of-Block service request is then issued to the I/O Distributor. Once the End-of-Block is acknowledged, the firmware returns to Idle :00.

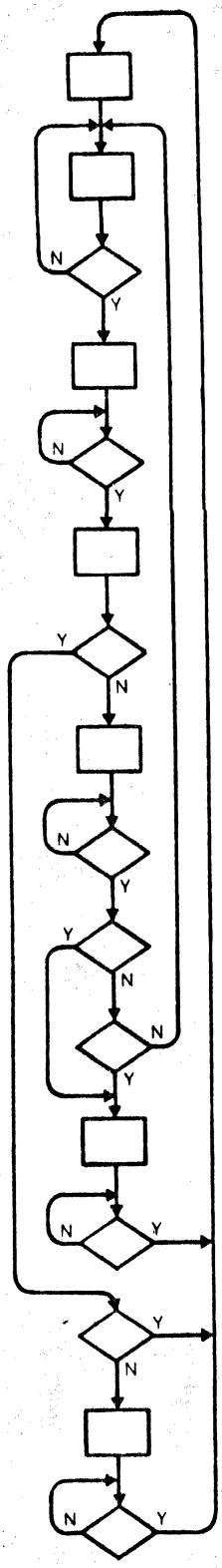
WAIT FOR SERVICE REQUEST :0E

The Wait-for-Service request firmware sequence is used when the computer has programmed a Peripheral to perform some function (such as taking a measurement) and then must wait for the Peripheral to complete the function. By sensing when the service request occurs and having the firmware issue an End-of-Block service request, the computer is saved the necessity of requesting status to determine when the Peripheral requires service.

ACCEPTOR HANDSHAKE :11

The Acceptor Handshake firmware sequence is used any time the IEC is a listener on the Interface Bus.

The Acceptor Handshake firmware sequence first signals the Interface Bus talker that the IEC is ready to accept data. The firmware sequence then waits for a Data Valid (DAV) over the Interface Bus. Once received, the data is first stored in the input data register. The firmware then checks if the operation is Unconditional I/O. If it is, the firmware checks further to see if interrupts are suppressed (FSUP = 1). If they are not, the firmware issued an End-of-Block service request to the I/O Distributor. Once the End-of-Block is acknowledged, the firmware returns to Idle :10. If the operation is Auto I/O or DMA, the firmware issues a data service request to the I/O Distributor. When the request is acknowledged, the firmware checks for byte count = 0 and then for an END remote message input (EOI = 1). If neither occurred, the firmware signals the Interface Bus talker to continue transferring data. If either ending condition occurred, the firmware issues an End-of-Block service request to the I/O Distributor. Once the End-of-Block is acknowledged, the firmware returns to Idle :10.




Idle - firmware loops in :10 until a Branch command word is received to initiate a firmware sequence.

Entry point :11
Send the ready for data signal to the Interface Bus talker. (NRFD = 0)

Is data available for input? (DAV = 1)

Signal the Interface Bus talker the data is accepted. (NDAC = 0, NRFD = 1)

Is the data transfer completed? (DAV = 0)

Acknowledge completion of data transfer. (NDAC = 1)

Is this operation Unconditional I/O? (FNT2 = 1)

Issue a data service request to the I/O Distributor.

Was the request acknowledged?

Is the data transfer byte count = 0?

Did the Interface Bus talker transfer the END remote message?

Issue an end-of-block service request to the I/O Distributor.

Was the request acknowledged?

Are the PICOPROCESSOR interrupt requests suppressed? (FSUP = 1)

Issue an end-of-block service request to the I/O Distributor.

Was the request acknowledged?

Figure 3-77b. Acceptor Firmware Sequence Flow Chart

### 3.3.10.5 Interface Description

The IEC implemented IEEE Interface Bus has two separate interfaces:

I/O Distributor to PICOPROCESSOR
INTERFACE BUS originating at the PICOPROCESSOR

The I/O Distributor to PICOPROCESSOR interface is the standard intelligent cable interface.

The Interface Bus, originating at the PICOPROCESSOR conforms to the requirements of the IEEE Specification. Refer to the document "IEEE Std. 488-1975" for a description of the Interface Bus.

The IEC is shipped with a 12 ft. (3.7 m) Interface Bus cable attached to the PICO-PROCESSOR. This cable is equipped with a male connector that conforms to the IEEE Specifications. Table 3-12 is the pin list for this cable.

Table 3-12. Interface Signal Assignments

| PICOPROCESSOR CONN | PIN | SIGNAL NAME | DESCRIPTION | INTERFACE BUS CONNECTOR PIN |
|---|---|---|---|---|
| P/J2 | 9 | DIO1 | DATA BIT 0 | 1 |
| P/J2 | 10 | DIO2 | DATA BIT 1 | 2 |
| P/J2 | 11 | DIO3 | DATA BIT 2 | 3 |
| P/J2 | 12 | DIO4 | DATA BIT 3 | 4 |
| P/J1 | 9 | EO1 | END OR INTERRUPT | 5 |
| P/J1 | 11 | DAV | DATA VALID | 6 |
| P/J1 | 12 | NRFD | NOT READY FOR DATA | 7 |
| P/J1 | 13 | NDAC | NOT DATA ACCEPTED | 8 |
| P/J1 | 14 | IFC | INTERFACE CLEAR | 9 |
| P/J1 | 1 | SQR | SERVICE REQUEST | 10 |
| P/J1 | 4 | ATN | ATTENTION | 11 |
| P/J2 | 3 | GND | SHIELD | 12 |
| P/J2 | 13 | DIO5 | DATA BIT 4 | 13 |
| P/J2 | 14 | DIO6 | DATA BIT 5 | 14 |
| P/J1 | 7 | DIO7 | DATA BIT 6 | 15 |
| P/J1 | 8 | DIO8 | DATA BIT 7 | 16 |
| P/J1 | 10 | REN | REMOTE ENABLE | 17 |
| P/J1 | 6 | GND | ISOLATER | 18 |
| P/J1 | 3 | GND | ISOLATER | 19 |
| P/J1 | 2 | GND | ISOLATER | 20 |
| P/J2 | 1 | GND | ISOLATER | 21 |
| P/J2 | 7 | GND | ISOLATER | 22 |
| P/J1 | 5 | GND | ISOLATER | 23 |
| P/J2 | 5 | GND | LOGIC GROUND | 24 |

3.3.10.3.6 Programming Example

The Assembler Language statements shown in the following example demonstrate a basic program for using the IEC with the Distributed I/O System. This example shows the various routines necessary to issue a program to a Digital Volt Meter (DVM) and to request input of a reading from the DVM. The demonstration code presents effectively two different operations with some routines used with both operations.

The first section of code is the parameter equates that define a series of constants used to assemble the various routines. The first five establish the device address and interrupt vector locations. The remaining equates select Intelligent Cable or Interface Bus operations.

```
*
* PARAMETER EQUATES FOR IEEE INTELLIGENT CABLE
*
INTBAS    EQU       :C0             INTERRUPT VECTOR BASE (STRAPPED ON I/O DISTRIBUTOR)
CHAN      EQU       3               IEEE CABLE IS PLUGGED INTO THIS CHANNEL
IOD       EQU       :F              ADDRESS BASE (STRAPPED ON I/O DISTRIBUTOR)
INTAD     EQU       CHAN%3+INTBAS   DATA INTERRUPT VECTOR ADDRESS FOR IEC
IECDA     EQU       IOD%3+CHAN%1    IEC DEVICE ADDRESS
*
SETMOD    EQU       :400            SET MODE COMMAND WORD BASE
REN       EQU       :20             REN BUS SIGNAL (REMOTE ENABLE)
ATN       EQU       :80             ATN BUS SIGNAL (ATTENTION)
BEGIN     EQU       :200            BRANCH COMMAND WORD BASE
SH        EQU       :1              SOURCE HANDSHAKE FIRMWARE SEQUENCE ADDRESS
AIDL      EQU       :10             ACCEPTOR IDLE FIRMWARE ADDRESS
AH        EQU       :11             ACCEPTOR HANDSHAKE FIRMWARE SEQUENCE ADDRESS
NOINIS    EQU       :C0             PROGRAMMED I/O WITHOUT INTERRUPTS OR FLAGS
CD        EQU       :800            CHARACTER DETECTION COMMAND WORD BASE
PS        EQU       :1000           PARITY STANDARDIZATION COMMAND WORD BASE
```

The following sections of code, interrupt vectored locations (output) and interrupt vectored locations (input), defines and fills the memory locations associated with the I/O Distributor channel to which the IEC is connected. There are six memory locations involved with each of these sections of code; however, these are the same six memory locations for both sections of code since all data service and End-of-Block service interrupts for the I/O Distributor channel where the IEC is connected are vectored to the same memory addresses. Each time the Automatic I/O operation must be switched from input to output or output to input, the interrupt vectored locations must be filled with the corresponding instruction information. When repeated input or output operations are performed, only the byte count and buffer address must be respecified at the start of the operation.

```
*
* INTERRUPT VECTORED LOCATIONS (OUTPUT)
*
          ABS       INTAD           INTERRUPT VECTORED LOCATIONS
          AOB       IECDA           AUTOMATIC OUTPUT BYTE INSTRUCTION
          DATA      -PRGCNT         NEGATIVE OF THE PROGRAMMING BYTE COUNT
          BAC       PRGBUF-1        ADDRESS OF THE PROGRAMMING BUFFER
          DATA      0               NOT USED
          JST       *$+1            CALL END-OF-BLOCK ROUTINE
          DATA      EOBLOC          ADDRESS OF END-OF-BLOCK ROUTINE
```

```
*
* INTERRUPT VECTORED LOCATIONS (INPUT)
*
          ABS       INTAD               INTERRUPT VECTORED LOCATIONS
          AIB       IECDA               AUTOMATIC INPUT BYTE INSTUCTION
          DATA      -RDGCNT             NEGATIVE OF THE READING BYTE COUNT
          BAC       RDGBUF-1            ADDRESS OF THE READING BUFFER
          DATA      0                   NOT USED
          JST       *$+1                CALL END-OF-BLOCK ROUTINE
          DATA      EOBLOC              ADDRESS OF END-OF-BLOCK ROUTINE
```

The following section of code is a subroutine used to output the contents of A register. A register is loaded prior to calling the subroutine. The subroutine waits until the IEC signals not busy before returning to the caller.

```
*
* SUBROUTINE TO OUTPUT ONE BYTE USING PROGRAMMED I/O WITHOUT INTERRUPTS
*
          ABS       :400 ONEBYT    ENT                          ENTRY POINT FOR
SUBROUTINE
          OTA       IECDA               STORE OUTPUT BYTE IN IEC
          LDA       =BEGIN+SH+NOINTS    BRANCH COMMAND WORD TO START IEC
          OTA       IECDA+1             SEND COMMAND WORD TO I/O DISTRIBUTOR
NOTYET    INA       IECDA+1             INPUT STATUS FROM IEC
          LRA       2                   ISOLATE THE BUSY FLAG
          JOS       NOTYET              LOOP IF IEC IS BUSY
          RTN       ONEBYT              RETURN TO CALLER IF IEC IS NOT BUSY
```

The following sections of code are a routine that outputs a program to the Digital Volt Meter. For use with this routine, the interrupt vectored locations must be defined for output. Both Automatic I/O and programmed I/O are used in performing this routine. Once the digital volt meter is programmed, the computer halts (see End-of-Block service).

```
*
* ADDRESS THE DIGITAL VOLT METER AS A LISTENER
*
DVMLIS    EQU       $
          LDA       =SETMOD+REN+ATN     ASSEMBLE SET MODE COMMAND WORD
          OTA       IECDA+1             SEND SET MODE TO ASSERT REN AND ATN TO IEC
          LAP       :3F                 UNLISTEN INTERFACE MESSAGE
          JST       ONEBYT              OUTPUT THE INTERFACE MESSAGE
          LAP       :5F                 UNTALK INTERFACE MESSAGE
          JST       ONEBYT              OUTPUT THE INTERFACE MESSAGE
          LAP       :29                 LISTEN ADDRESS OF THE DVM
          JST       ONEBYT              OUTPUT THE MESSAGE
          LDA       =SETMOD+REN         SET MODE COMMAND WORD TO REMOVE ATTENTION
          OTA       IECDA+1             SEND SET MODE COMMAND WORD TO IEC

*
* PROGRAM THE DIGITAL VOLT METER
*
          LDA       =BEGIN+SH           BRANCH COMMAND WORD TO START IEC
          OTA       IECDA+1             SEND COMMAND WORD TO I/O DISTRIBUTOR
          EIN                           ENABLE INTERRUPTS
          JMP       $                   WAIT FOR INTERRUPT TO END-OF-BLOCK SERVICE
```

```
*
*  AUTOMATIC I/O BYTE COUNT AND BUFFER ADDRESS
*
PRGCNT    EQU       6                  LENGTH OF DVM PROGRAM
PRGBUF    TEXT      'AFTLO.'           DVM PROGRAM IN BUFFER
```

The following sections of code are a routine that accepts a reading from the Digital Volt Meter. For use with this routine, the interrupt vectored locations must be defined for input. Both Automatic I/O and programmed I/O are used in performing this routine. Once the reading is input by the digital volt meter, the computer halts (see End-of-Block service). The reading can be displayed or printed out by accessing the reading buffer.

```
*
* ADDRESS THE DIGITAL VOLT METER AS THE TALKER
*
DVMTLK    EQU       $
          LDA       =SETMOD+REN+AIN    ASSEMBLE SET MODE COMMAND WORD
          OTA       IECDA+1            SEND SET MODE TO ASSERT REN AND ATN TO IEC
          LAP       :3F                UNLISTEN INTERFACE MESSAGE
          JST       ONEBYT             OUTPUT THE INTERFACE MESSAGE
          LAP       :5F                UNTALK INTERFACE MESSAGE
          JST       ONEBYT             OUTPUT THE INTERFACE MESSAGE
          LAP       :49                TALK ADDRESS OF THE DVM
          JST       ONEBYT             OUTPUT THE MESSAGE
          LDA       =BEGIN+AIDL        BRANCH COMMAND WORD TO ASSERT NRFD
          OTA       IECDA+1            SEND COMMAND WORD TO I/O DISTRIBUTOR
          LDA       =SETMOD+REN        SET MODE COMMAND WORD TO REMOVE ATTENTION
          OTA       IECDA+1            SEND SET MODE COMMAND WORD TO IEC

*
* ACCEPT A READING FROM THE DIGITAL VOLT METER
*
          LDA       =BEGIN+AH+CD+PS    BRANCH COMMAND WORD TO START IEC
          OTA       IECDA+1            SEND COMMAND WORD TO I/O DISTRIBUTOR
          EIN                          ENABLE INTERRUPTS
*         JMP       $                  WAIT FOR INTERRUPT TO END-OF-BLOCK SERVICE
*
* AUTOMATIC I/O BYTE COUNT AND BUFFER ADDRESS
*
RDGCNT    EQU       20                 LENGTH OF BUFFER FOR DVM READING
RDGBUF    RES       20                 READING BUFFER
```

After all Automatic I/O data transfers are completed, the IEC vectors the last I/O interrupt to the End-of-Block service location. The subroutine called by this interrupt can be used to initiate processing of data transferred, perform error analysis, etc. The demonstration code simply halts.

```
*
* END-OF-BLOCK SERVICE
*
EOBLOC    ENT                          END OF BLOCK ENTRY POINT
          HLT
```

### 3.3.11 Line Printer Intelligent Cable (Dataproducts)

The Line Printer Intelligent Cable 14631-13 controls the output of data from an LSI Family Computer to a Dataproducts Model 2230/2260 (or other plug-compatible) Line Printer.

#### 3.3.11.1 Line Printer System Description

The Line Printer System is a data output system that displays the data in printed, hard-copy, form. The Line Printer contains all logic necessary for the actual printing of the data characters. In addition, the Line Printer performs a limited amount of "housekeeping" in controlling the paper supply and limiting line length. All other format control for the Line Printer is provided by the computer software.

Depending on the configuration of the Dataproduct Line Printer, three basically different sets of system operation characteristics are available.

If the Line Printer does not have a Vertical Format Unit (VFU) and standard Dataproducts wiring is used, the general characteristics of the Line Printer system are:

- Data is accepted as 7-bit ASCII with an unused eighth bit. The eighth may be set to a 1 or a 0.
- Format control, line termination and paper feed, is provided by software embedding specified format control characters in the data stream transferred to the line printer.

Configured in this manner, the Dataproducts Line Printer operates similar to but not software compatible with the Centronics Line Printer Intelligent Cable.

If the Line Printer does not have a Vertical Format Unit (VFU), a wiring option may be performed on the Dataproducts Line Printer to provide for both carriage return and line feed when the Line Printer detects the carriage return character. This option provides for full software compatibility with the Centronics Line Printer Intelligent Cable. The general characteristics in this configuration are:

- Data is accepted as 7-bit ASCII with an unused eighth bit.
- Format control is provided by software embedding specified format control characters in the data stream transferred to the Line Printer.

If the Line Printer has a Vertical Format Unit (VFU), the general characteristics of the Line Printer system are:

- The data transferred in accepted as a 7-bit ASCII character when the eighth bit is a 0.
- The data transferred is accepted as a 7-bit VFU control character when the eighth bit is a 1.
- Format control is provided by software embedding format control characters or a combination of VFU control characters and format control characters in the data stream transferred to the Line Printer.

For details of format control and VFU operation refer to Section 3.3.11.3.3, Format Considerations in this manual. The Line Printer Intelligent Cable manages the actual transfer of data. Figure 3-78 shows the Line Printer system configuration.

### 3.3.11.2 Specifications

Cable Length (Nominal), 12.5 ft (3.8m)

I/O Distributor to PICOPROCESSOR, 10.5 ft (3.2m)

PICOPROCESSOR to Line Printer, 1.5 ft (.46m)

Data Type, 7-bit output

Operating Mode, Simplex

Command Output, Single command format

Status Input, 6-bit format

Standard Channel Number, 7 (Device Address Field :FE)

Standard Data Service Vector Address, :F8

Standard End-of-Block Service Vector Address, :FC

Strapping Required, None

### 3.3.11.3 Software Considerations

The Line Printer Intelligent Cable conforms to the general I/O Distributor programming philosophy described in Section 2. The Line Printer Intelligent Cable outputs data in 7-bit parallel bytes. The eighth bit specifies the associated character as data (0) or VFU control (1). The Line Printer Intelligent Cable does not input data to the computer.

The computer instructions for both data and control transfers are unchanged from standard usage.

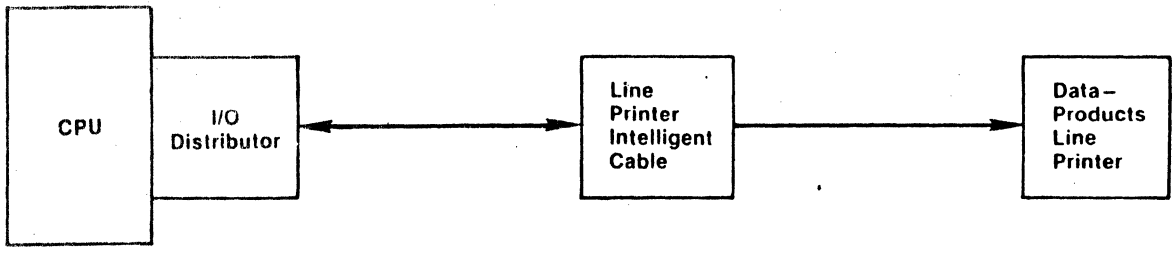



Figure 3-78. Dataproducts Line Printer System Configuration

### 3.3.11.3.1 Line Printer Intelligent Cable Command Word Set

The Command Word Set used with the Line Printer Intelligent Cable consists of three of the standard command words described in Section 2. The following is a list of the command words and applicable hexadecimal word skeletons used with the Line Printer Intelligent Cable. The hyphens represent a field that must be specified.

Reset :0100
Branch :02-0
Branch, Disable DMA :82-0

The Branch, Disable DMA command word does not affect the operation of the standard I/O Distributor and is treated the same as a Branch command word by the standard I/O Distributor.

### 3.3.11.3.2 Branch Address Field Description

The Line Printer Intelligent Cable PICOPROCESSOR uses the standard 4-bit branch address field. The following is the allowed entry point to the firmware.

:1 - Start Printing. This entry point to the firmware is use to start the printing operation.

### 3.3.11.3.3 Status Byte Description

The Line Printer Intelligent Cable's PICOPROCESSOR provides three bits of status information for the computer. The status bits are arranged in a single status byte. The individual status bits are input to the PICOPROCESSOR from the line printer or derived from the PICOPROCESSOR operation. The status register in the PICOPROCESSOR also operates as the sense register for firmware sequencing control.

The bit assignments for the status byte are shown in Figure 3-79. The following describes the conditions reported by each bit.

Bit 0 and Bit 2 - Demand Data. The Demand Data bit is a logical 1 when the Line Printer is able to accept data from the PICOPROCESSOR. The Demand Data bit is a logical 0 while the line printer stores the transferred data and during the print operation. Demand Data is also a logical 1 any time the line printer is de-selected (off-line).

Bit 3 - Sense Interrupt. The Sense Interrupt bit is a logical 1 whenever the PICOPROCESSOR is inputting a data service request to the I/O Distributor.

Bit 5 - Online. The Online bit is a logical 0 anytime the Line Printer is properly cabled to the PICOPROCESSOR, powered, and ready for a print operation.

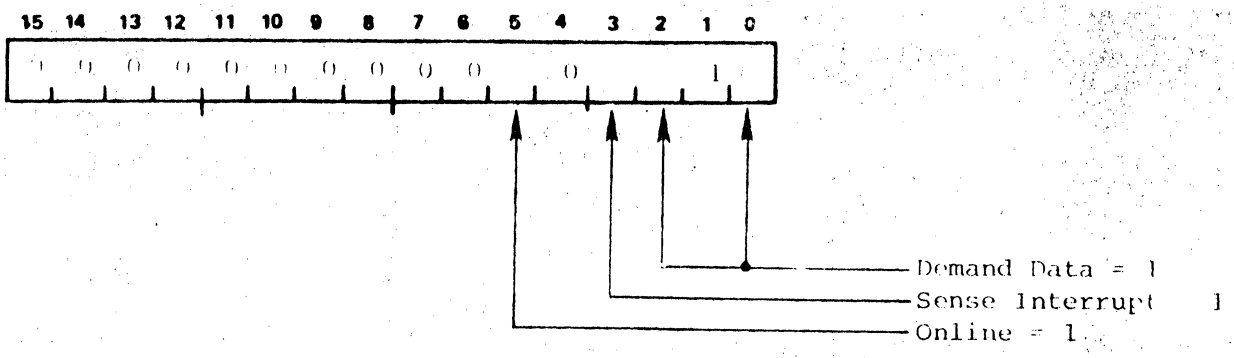



Figure 3-79. Status Byte Configuration

3.3.11.3.4 Format Considerations

The Line Printer Intelligent Cable controls format and vertical paper movement on the Dataproducts Line Printers by one of two methods:

- By insertion of format control characters into the data stream.
- By commands, inserted in the data stream, to the optional Vertical Format Unit (VFU) in the Line Printer.

The format control characters consist of three 7-bit characters recognized by the Line Printer as format control characters.

Paper Feed :0A, This character causes the paper to advance one line vertically.

Forms Feed :0C, This character causes the paper to advance to the top of the next page.

Carriage Return :0D, This character initiates the printing operation. The paper is not advanced by the Carriage Return character.

Dataproducts Line Printers without a vertical format unit can be operated with Centronics Line Printer software provided the Data Products Carriage Return is internally modified to include a single space paper feed. Refer to the Dataproducts hardware user's manual for details on the required modification.

The VFU commands consist of the appropriate 7-bit character output simultaneous with the paper control signal. In the Line Printer Intelligent Cable, the paper control signal is determined by the state of data bit 7 of the output data byte. Figure 3-80 illustrates the format of a complete Line Printer data word in the computer before output to the line printer. The paper control bits are bits 7 and 15. If the paper control bit is a logical 0, the Line Printer accepts the associated character as data for printing. If the paper control bit is a logical 1, the Line Printer accepts the associated character as a VFU control character.

Two different types of VFU commands are accepted by the line printer. These commands are:

- Move paper (slew) for the specified number of lines, 0 through 16.
- Move paper (slew) until a hole is encountered in the specified VFU tape channel 0 through 11.

Table 3-13 lists the various VFU control characters. Data bits 5 and 6 are ignored by the Line Printer when performing a VFU operation.

Data prepared for printing using the VFU control characters cannot be printed on a Line Printer without a Vertical Format Unit. Data prepared for printing using the vertical control characters can be printed on a Line Printer with the Vertical Format Unit provided data bit(s) 7 (and 15) is always set to 0.

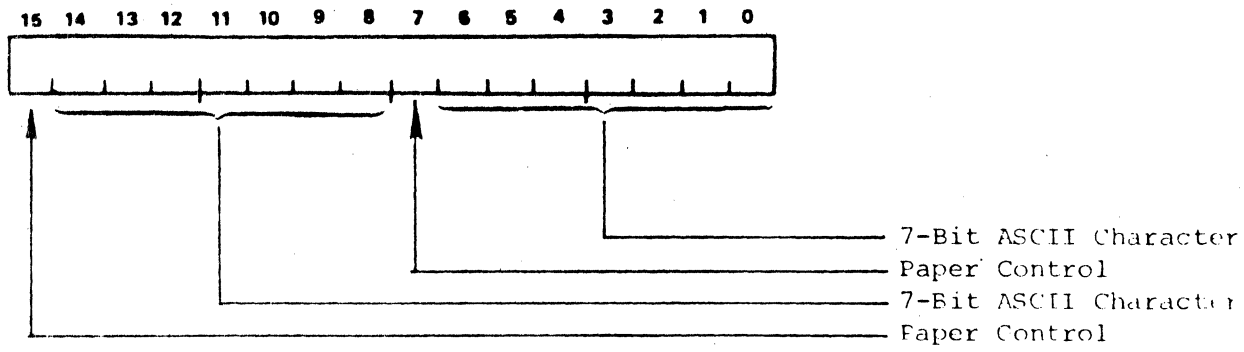



Figure 3-80. Line Printer Data Word Format

Table 3-13. Vertical Format Unit Control Characters

| VFU Character | Number of Lines Slewed | VFU Character | VFU Tape Channel Selected |
|---|---|---|---|
| :90 | 0 | :80 | 0 |
| :91 | 1 | :81 | 1 |
| :92 | 2 | :82 | 2 |
| :93 | 3 | :83 | 3 |
| :94 | 4 | :84 | 4 |
| :95 | 5 | :85 | 5 |
| :96 | 6 | :86 | 6 |
| :97 | 7 | :87 | 7 |
| :98 | 8 | :88 | 8 |
| :99 | 9 | :89 | 9 |
| :9A | 10 | :8A | 10 |
| :9B | 11 | :8B | 11 |
| :9C | 12 | | |
| :9D | 13 | | |
| :9E | 14 | | |
| :9F | 15 | | |

### 3.3.11.3.5 Data Transfer Timing Considerations

The transfer of data to a line printer such as the Dataproducts model 2230/2260 is typified by long periods when data service requests are not made (for example, approximately 0.2 seconds at 300 lines per minute). This inactive period is followed by a burst of up to 136 data requests plus additional data service requests for paper control characters. With the Dataproducts Line Printers, this burst of data must be transferred within 2.5 ms to maintain the line per minute speed. This requires a 55K byte data transfer rate if full 136 character lines are being printed. When interfaced through a DMA I/O Distributor, the rated printing speed is easily maintained. With the standard I/O Distributor, the additional variables of interrupt priorities, interrupt latency, and interrupt processing time makes it impossible to determine if such a data burst can be transferred in the time allowed. However, the result of failing to make a complete burst transfer in the time allowed is not a loss of data but rather a slight slowing of the continuous printing speed.

### 3.3.11.4 Operating Sequence

The operation of the Line Printer System is controlled by software direction of the Line Printer with operation sequencing provided by software selection of the appropriate PICOPROCESSOR firmware sequence.

### 3.3.11.4.1 Software Sequence

All online operations of the Line Printer System are initiated by software issuing a Branch command word to the PICOPROCESSOR. Prior to the issuance of the command, the software must determine the present conditions of the Line Printer. If status was requested and stored at the termination of the previous operation, the software can examine the stored status bits. If an End-of-Block was expected and not received, an operation is in progress. Once the necessary housekeeping is completed, the software can issue the Branch command word.

### 3.3.11.4.2 Firmware Sequences

Once initiated, the PICOPROCESSOR'S firmware sequences assumes control of the I/O operation until terminating conditions are sensed. The following describes the firmware sequenses flow charted in Figure 3-81.

IDLE (:0)

The PICOPROCESSOR loops in Idle anytime a firmware sequence is not being performed. The final step of all firmware sequences is a return to Idle. The firmware must be in Idle to accept a Branch command word.

START PRINTING (:1)

The firmware sequence has a single entry point at :1. The firmware first makes a system check for the Line Printer being powered and online. Failure of the system check results in the printing operation being aborted. The firmware issues an End-of-Block service request to the I/O Distributor to indicate terminating conditions were detected. The End-of-Block service request is held until acknowledged, at which time, the firmware returns to Idle.

After the system check the firmware checks to see if the Line Printer is requesting for a data character. If so, the firmware issues a data service request to the I/O Distributor. After the request is acknowledged, the data character provided from the computer is output to the Line Printer. The firmware then waits for the Line Printer to accept the data character.

If the byte count did not go to zero with that data character, the firmware then returns to the system check in preparation for the next data transfer. When the byte count goes to zero, the firmware issues an End-of-Block service request to the I/O Distributor to indicate terminating conditions were detected. The End-of-Block service request is held until acknowledged, at which time, the firmware returns to Idle.

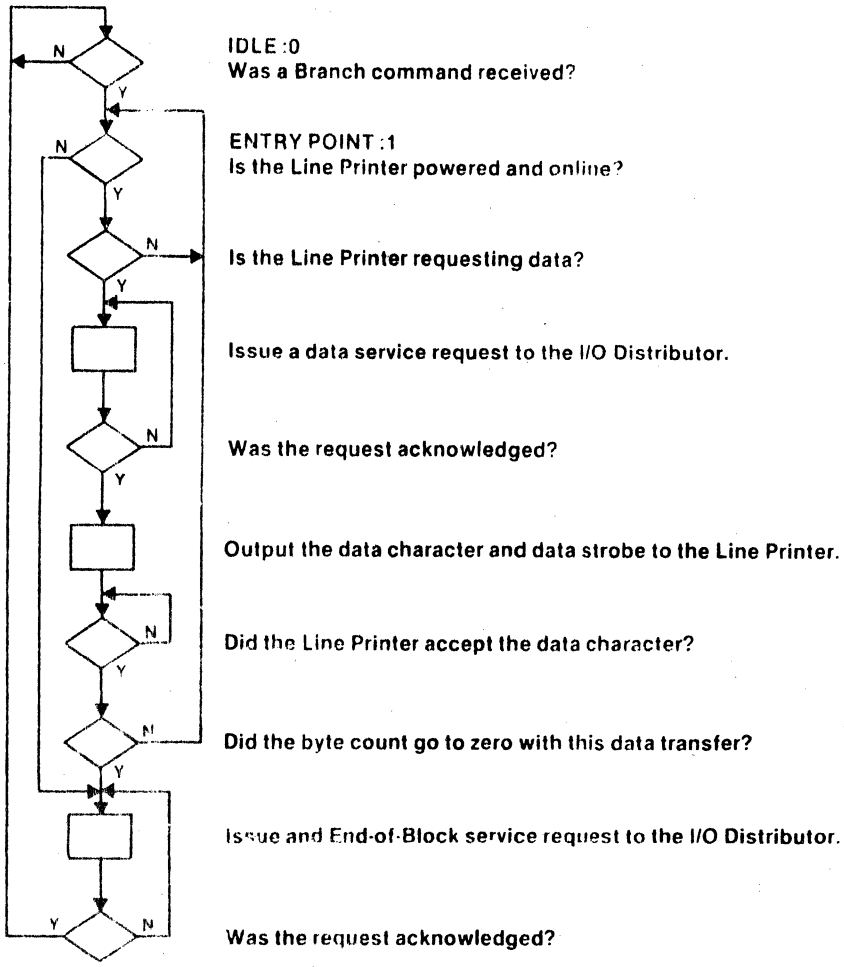



Figure 3-81. Firmware Sequence-Line Printer Intelligent Cable

3.3.11.5 Interface Description

The Line Printer Intelligent cable has two interfaces:

I/O Distributor to PICOPROCESSOR

PICOPROCESSOR to Line Printer

The I/O Distributor to PICOPROCESSOR interface is the standard Intelligent Cable interface.

The PICOPROCESSOR to Line Printer interface is carried over a 48-conductor flat ribbon cable. This device cable is supplied with a Winchester MRAC 50P-JTCH device mating connector. Figure 3-82 is the PICOPROCESSOR to Line Printer interface overview. Table 3-14 lists the signal assignments for the device cable.

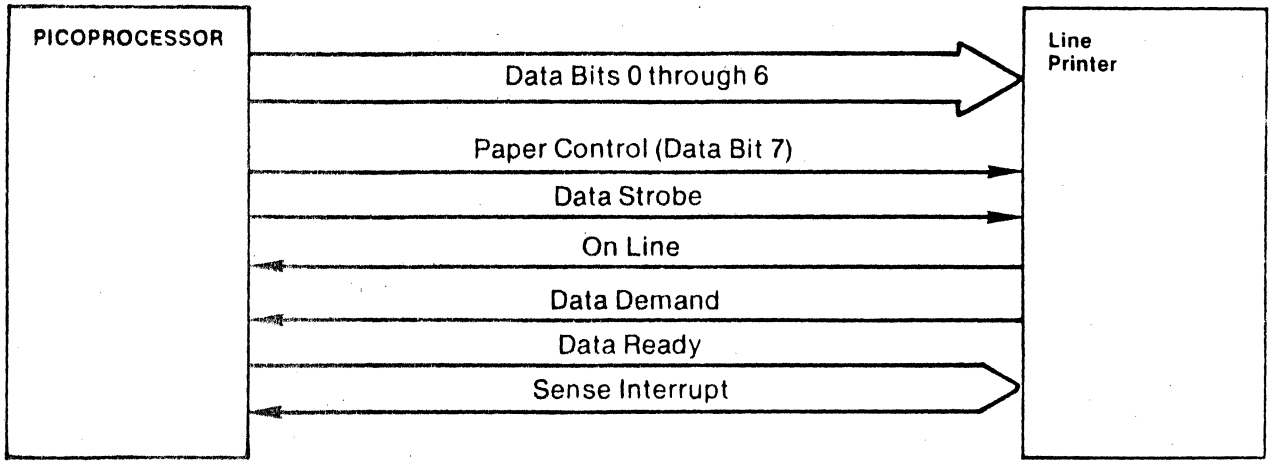


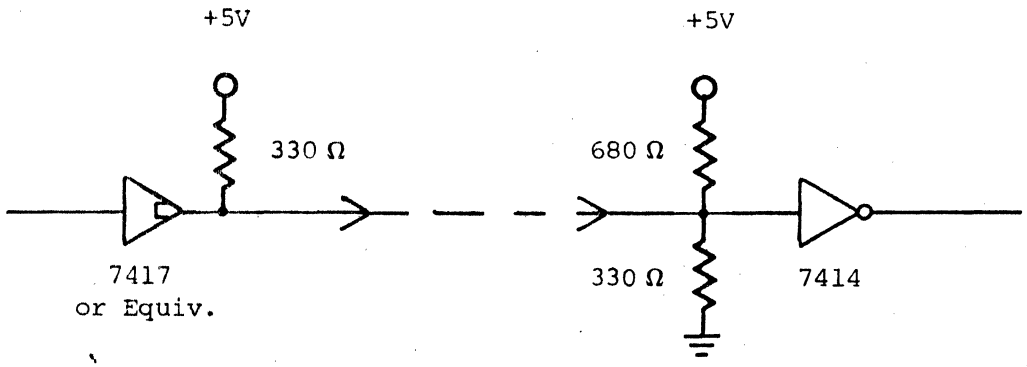

Figure 3-82. Interface Overview

Table 3-14. Interface Signal Assignments

| PICOPROCESSOR CONNECTOR | PICOPROCESSOR PIN | DESCRIPTION | DEVICE CONNECTOR PIN NUMBER |
|---|---|---|---|
| J5 | 6 | Data Bit 0 | B |
| J5 | 10 | Ground | C |
| J4 | 3 | Ground | D |
| J4 | 9 | Ground | D |
| J4 | 2 | Data Demand | E |
| J4 | 4 | Data Demand | E |
| J5 | 11 | Data Bit 1 | F |
| J4 | 10 | Ground | J |
| J5 | 5 | Data Bit 2 | L |
| J4 | 11 | Ground | N |
| J5 | 12 | Data Bit 3 | R |
| J4 | 12 | Ground | T |
| J5 | 4 | Data Bit 4 | V |
| J4 | 13 | Ground | X |
| J5 | 13 | Data Bit 5 | Z |
| J4 | 14 | Ground | b |
| J4 | 6 | Data Strobe | j |
| J4 | 15 | Ground | k |
| J4 | 16 | Ground | m |
| J5 | 3 | Data Bit 6 | n |
| J5 | 14 | Paper Control (Data Bit 7) | p |
| J5 | 9 | Ground | s |
| J5 | 7 | OnLine | y |
| J6 | 1 | Ground | AA |
| J4 | 1 | Sense Interrupt | Jumpered Together |
| J4 | 8 | Data Ready | Jumpered Together |

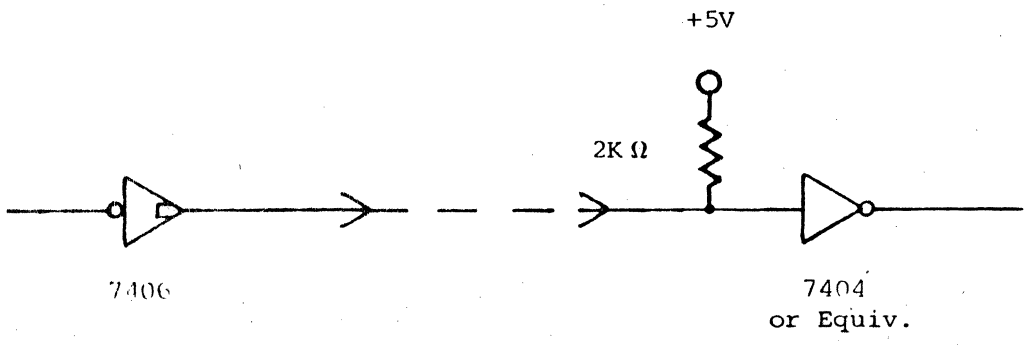
Output Logic

The output drivers for the data and control lines are open-collector TTL buffers capable of sinking up to 32 mA. The termination resistors in the Line Printer are optional for high noise immunity and are not provided by Dataproducts if unspecified when the Line Printer is ordered. If termination resistors are desired, specify the values shown.




Input Logic

The input receivers use standard TTL logic levels, 0 to +5 volts, and present one TTL load with a 2K OHM pull-up resistor to +5 volts.



3.3.11.6 Programming Example

The Assembler Language statements shown in Figure 3-83 demonstrate one method for using the Distributed I/O System.

The demonstration code falls into three distinct parts, interrupt vectored locations, I/O initiation and End-of-Block service.

INTERRUPT VECTORED LOCATIONS

The first part of the code defines and fills the standard locations associated with the I/O Distributor channel to which the Line Printer Intelligent Cable is connected. There are six locations involved. Two, the byte count and the buffer location, are dynamic for each I/O operation. The remaining words could be fixed at program load time.

I/O INITIATION

The second part of the code is used to transfer one physical record each time the mainline program executes the instruction, JST LPRINT.

It is assumed the calling program has previously set up the following two words:

| | |
|---|---|
| BYTCNT | Number of bytes to be transferred |
| BUFADD | Word address of record buffer |

The demostration code converts this information into the form required by an Automatic I/O instruction, a negative byte count and the byte address of the buffer minus one. Once these computations are stored in the interrupt vectored locations, the PICO-PROCESSOR is sent the command word, Begin at Branch Address :1.

No attempt is made in the demonstration code to overlap record transfer with CPU activity. Instead, a dead loop is used, JMP $, until the End-of-Block interrupt vector is received. At this point control passes to the subroutine labelled EOB.

END-OF-BLOCK SERVICE

The End-of-Block subroutine can be used to perform error analysis, retry, etc. The demonstration code simply obtains device status from the Line printer intelligent cable and passes the status to the mainline program through the A register.

```
*
* INTERRUPT VECTORED LOCATIONS
*
LPINT     EQU       :F8         STANDARD LINE-PRINTER INTERRUPT
LPDEVA    EQU       :FE         STANDARD DEVICE ADDRESS

          ABS       LPINT       INTERRUPT VECTORED LOCATIONS
          AOB       LPDEVA      AUTO OUTPUT BYTE
          RES       1,0         TO BE FILLED WITH BYTE COUNT
          RES       1,0         TO BE FILLED WITH BUFFER ADDRESS-1
          DATA      0           NOT USED
          JST       *$+1        CALL END-OF-BLOCK ROUTINE
          DATA      EOB         ADDRESS OF END-OF-BLOCK ROUTINE

*
* I/O INITIATION
*
LPRINT    ENT                   ENTRY POINT FOR LP DRIVER
          LDA       BYTCNT      PHYSICAL RECORD BYTE COUNT
          NAR                   AOB INSTRUCTION NEEDS NEGATIVE
          STA       LPINT+1     NEGATIVE BYTE COUNT IN AOB INSTRUCTION
          LDA       BUFADD      ADDRESS (WORD) OF BUFFER
          LLA       1           AOB INSTRUCTION NEEDS BYTE ADDRESS
          SAI       1           STARTS AT -1
          STA       LPINT+2     PUT THIS ADDRESS IN AOB INSTRUCTION
          LDA       =:0210      COMMAND WORD TO START PICOPRESSOR
          OTA       LPDEVA+1    SEND COMMAND WORD TO PICOPROCESSOR
          JMP       $           WAIT FOR END-OF-BLOCK

*
* END-OF-BLOCK SERVICE
*
EOB       ENT                   END-OF-BLOCK INTERRUPT SUBROUTINE
          INA       LPDEVA+1    INPUT STATUS
          RTN       LPRINT      RETURN TO CALLER WITH
                                STATUS IN A REGISTER
```

Figure 3-83. Programming Example - Line Printer

### 3.3.12 32-Bit General-Purpose Intelligent Cable

The 32-Bit General-Purpose (G-P) Intelligent Cable allows the interfacing of most low- to medium-speed peripheral or special purpose input/output devices to an LSI Family Computer without the need to design and implement special interface logic. The 32-Bit G-P Intelligent Cable's word size is selectable in 8-bit increments and any device word size up to 32-bits, input and/or output can be accommodated.

The 32-Bit G-P Intelligent Cable is designed to transfer data using a two-wire "handshake" interface discipline. Other interface disciplines are accommodated using a combination of computer software and operational sequencing control.

#### 3.3.12.1 Typical Interface System Descriptions

The 32-Bit G-P Intelligent Cable is able to interface to peripheral devices in a number of control configurations.

Figure 3-84 illustrates both the simplex input and simplex output configuration. This type of configuration is used with devices whose operation is restricted to input only or output only.

When the peripheral device is capable of both input and output, the half-duplex configurations shown in Figures 3-85 and 3-86 are used. If simultaneous input and output are required, a full-duplex configuration with two 32-Bit G-P intelligent cables must be used.

The 32-Bit G-P Intelligent Cable is able to output a tag signal to specify when a data transfer consists of control information or data information. Through the use of this capability, several similar simplex or half-duplex peripheral devices can be bus connected to a single 32-Bit G-P Intelligent Cable. This type of configuration is illustrated in Figure 3-87.

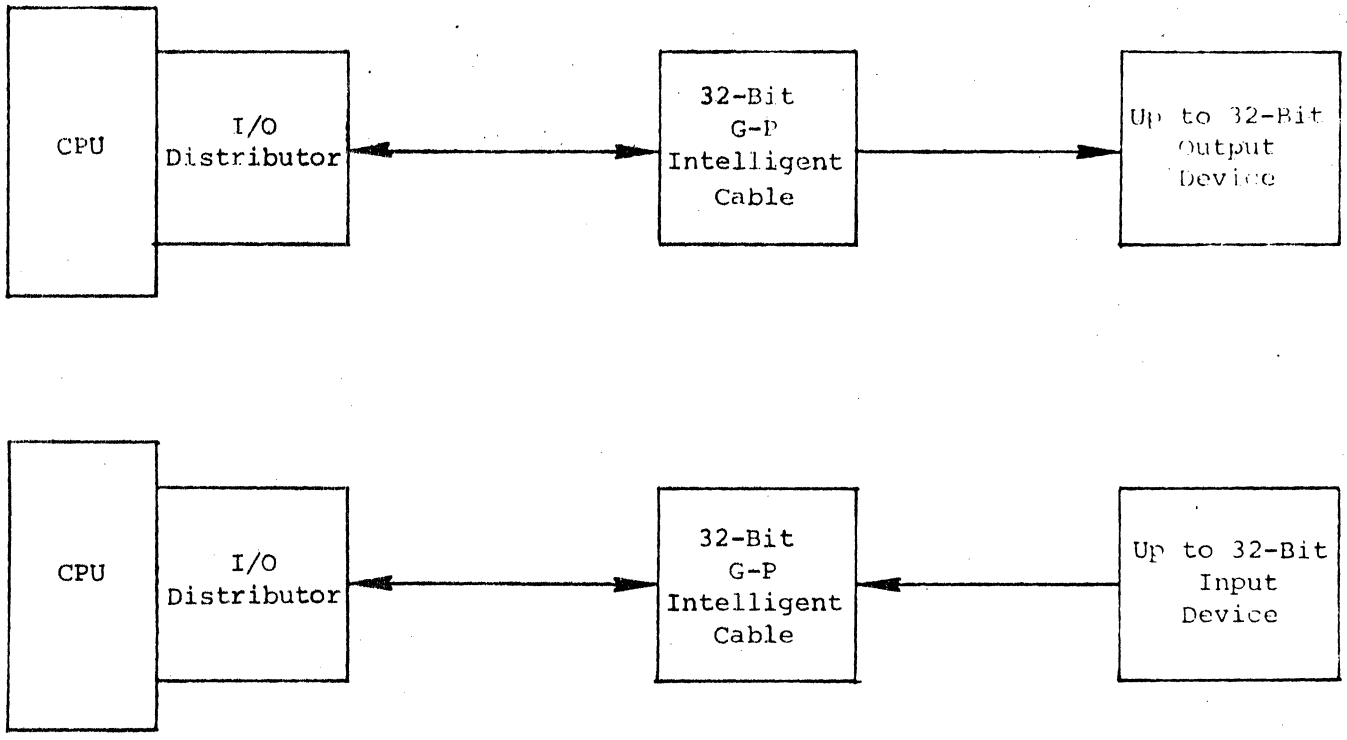



Figure 3-84. Simplex Device System Configuration

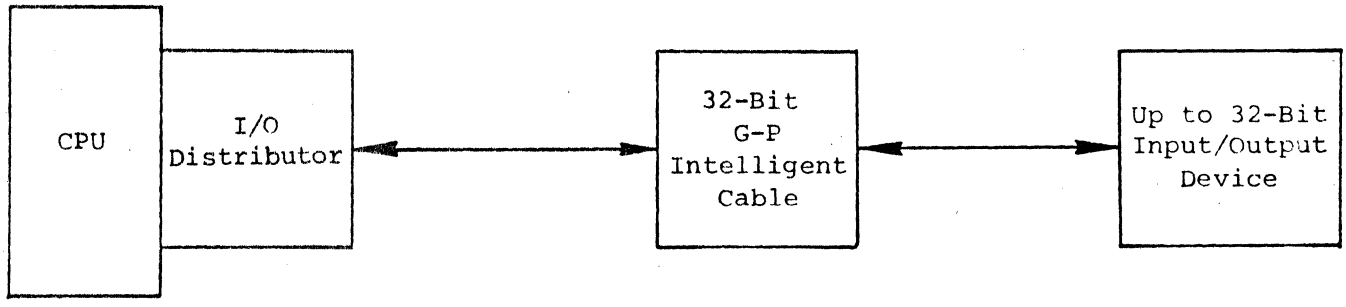



Figure 3-85. Half-Duplex Device System Configuration

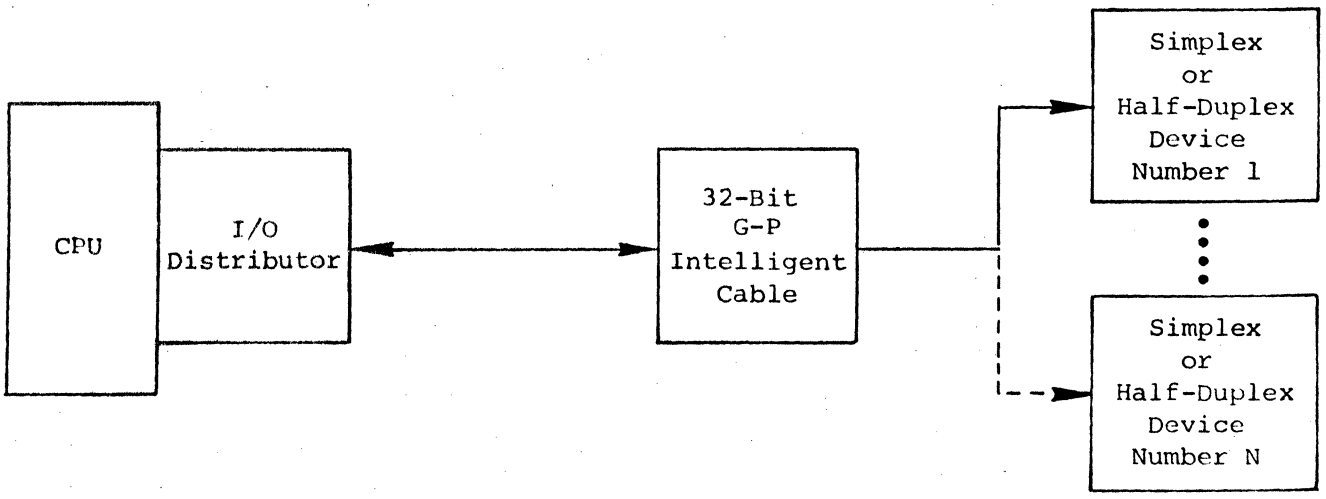



Figure 3-86. Interface Bus (Multi-Device) System Configuration

3.3.12.2 Specifications

Cable Length (nominal), 12.5 ft (3.8m)

I/O Distributor to PICOPROCESSOR, 10.5 ft. (3.2m)

PICOPROCESSOR TO DEVICE 1.5 ft (0.46m)

Data types . . . . . Selectable 8, 16, 24, or 32-bit parallel data words
Output with storage
Input without storage

Operating Modes . . . . . Simplex and half-duplex

Command Output . . . . . Single command word format with 4 mode bits

Status Input (PICOPROCESSOR operational). . . . . 8-bit format

I/O Disciplines . . . . . Two-wire "handshake"

Other I/O disciplines are accommodated through a combination of computer software and operational sequencing control.

Data Modes:

Data output . . . . . 8, 16, 24, or 32-bits
Data input . . . . . 8, 16, 24, or 32-bits
Command output . . . . . 8, 16, 24, or 32-bits
Device status input . . . . . 8, 16, 24, or 32-bits

Interface Logic . . . . . See Section 3.3.12.5

Standard Channel Number . . . . . None assigned

Standard Data Service Vector Address . . . . . None assigned

Standard End-of-Block Service Vector Address . . . . . None assigned

Bandwidth

Table 3-15 contains listings of the approximate maximum word transfer rate bandwidths. The listed maximums apply to a single 32-Bit G-P Intelligent Cable transferring data. All other peripherals must be in Idle, and the software in a minimum time wait loop.

Table 3-15. Maximum Word Transfer Rates

| CPU Type | I/O Distributor Type | Core 980 Memory | | | |
|---|---|---|---|---|---|
| | | 8-Bit Words | 16-Bit Words | 24-Bit Words | 32-Bit Words |
| LSI-2/60 | Standard | 86 | 43 | 29 | 21.5 |
| LSI-2/20 | Standard | 86 | 43 | 29 | 21.5 |
| LSI-2/10 | Standard | 43 | 21.5 | 14 | 11 |
| LSI-23/05 | Standard | 32 | 16 | 11 | 8 |
| LSI Family | DMA | 225 | 112 | 75 | 55 |

### 3.3.12.3 Software Considerations

The 32-Bit G-P Intelligent Cable conforms to the general I/O Distributor programming philosophy described in Section 2.

The 32-Bit G-P Intelligent Cable provides storage on the PICOPROCESSOR for a 32-bit word during output operations. During input, the device must present the data word for the full time required for the input transfers. The input or output word size used is software selectable in 8-bit increments at 8, 16, 24, and 32 bits. A separate data transfer is required for each eight bits of the input/output word.

The computer instructions for both data and control transfers are unchanged from standard usage. Peripheral devices that use the two wire "handshake" interface discipline can make use of the Auto I/O programming. Pulse and other interface disciplines can be accommodated by direct software driven (programmed) I/O control.

#### 3.3.12.3.1 32-Bit G-P Intelligent Cable Command Word Set

The command word set used with the 32-Bit G-P Intelligent Cable consists of 11 of the standard command words described in Section 2. The following is a list of the command words and applicable hexadecimal word skeletons used with the 32-Bit G-P Intelligent Cable. The command words shown in italics cannot be used with standard I/O Distributors Model 14629-14 and Model 14629-18. Each "n" represents a field that must be specified.

| | |
|---|---|
| Reset | :0100 |
| Branch | :02n0 |
| Set Mode | :040n |
| *Branch, Set Mode | :06nn |
| Branch, Detect CR | :0An0 |
| Branch, Set Mode, Detect CR | :0Enn |
| *Branch, Detect Special Character* | *:2An0* |
| *Branch, Set Mode, Detect Special Character* | *:2Enn* |
| *Load Special Character* | *:80nn* |
| Branch Disable DMA | :82n0 |
| Branch, Set Mode, Disable DMA | :86nn |

The Branch, Disable DMA and Branch, Set Mode, Disable DMA command words do not affect the operation of the standard I/O Distributor and are treated the same as the corresponding Branch or Branch, Set Mode command words by the standard I/O Distributor.

The character detection feature (Detect CR and Detect Special Character) should be used with caution when operating 16-, 24-, or 32-Bit input devices. Each 8-bit partial word transferred to the I/O Distributor is scanned and unwanted character detection can occur.

The parity standardization feature command words are not listed and are considered unusable.

*The Branch, Set Mode command word can be used when the operational sequence is in Idle :00 or :10; otherwise, separate Branch and Set Mode command words should be used.

3.3.12.3.2 Mode Field Description

The 32-Bit G-P Intelligent Cable PICOPROCESSOR accepts four mode bits for additional control of the peripheral device and PICOPROCESSOR operation. The Set Mode command words used with the 32-Bit G-P Intelligent Cable are the standard forms that provide a 4-bit mode field. Figure 3-87 shows the configuration of the Set Mode command words. The functions controlled by each of the mode bits are as follows:

Bit 0 and 1 - Word Size. The Word size bits are used to select the PICOPROCESSOR's input/output word size. Four word sizes are available; 8, 16, 24 and 32 bits. Refer to Figure 3-87 for the selection values. Mode bits 0 and 1 are not output to the device.

Bit 2 - Data/Command. Bit 2 is normally defined as the Data/Command bit for use when more than one type of data is being transferred between the computer and the peripheral device(s) via the 32-Bit G-P Intelligent Cable. The Data/Command bit can be used as follows:

- To output a device address when more than one peripheral device is attached to a single 32-Bit G-P Intelligent Cable.
- To output a program such as control settings for the front panel to a peripheral device.
- To accept status directly from the peripheral device.

The standard definitions for Bit 2 are Data = logical 1, Command = logical 0. Bit 2 is not used internally by the PICOPROCESSOR and may be redefined to suit the needs of the attached device.

Bit 3 - Input/Output. The Input/Output bit is used as the most significant address bit for the PICOPROCESSOR firmware. The logical 1 selects the input firmware sequences and the logical 0 selects the output firmware sequences. Bit 3 is output to the device along with being used internally by the PICOPROCESSOR.

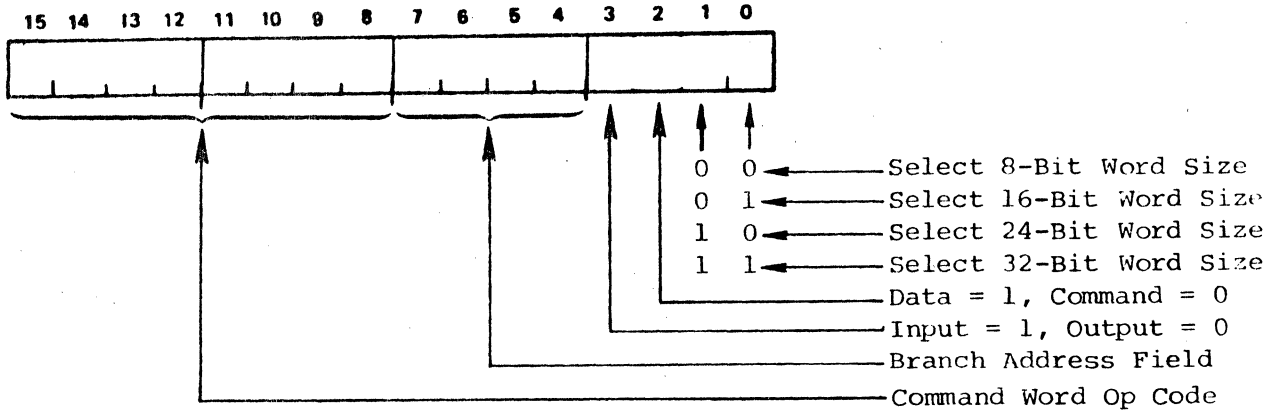



Figure 3-87. Set Mode Command Word Format

3.3.12.3.3 Status Byte Description

The 32-Bit G-P Intelligent Cable PICOPROCESSOR provides eight bits of status information for the computer. The status bits are arranged in a single status byte. Five of the individual status bits are input to the PICOPROCESSOR from the peripheral device and the remaining three bits are derived from the PICOPROCESSOR operation.

Status bits 0 through 4 are sampled at the appropriate times by the PICOPROCESSOR operation sequence. The state of a status bit, when sampled, is used to direct the operation sequence. The bit assignments for the status byte that conforms to operation sequence use are shown in Figure 3-88. These status bit assignments can be redefined to conform to actual use by the peripheral. However, any redefinition of a status bit should be approximately the same as the definition presented here to assure proper sequencing. For example, the Power status bit could be redefined as device ready, both representing a state of preparation for operation. The actual providing of inputs to status bits 0 through 3 is dependent on the peripheral device used and the device's interface discipline requirements. Refer to the Interface Description (Section 3.3.12.5) for the connection and termination requirements for each of the status lines.

The status byte reported to the computer is the logical complement of the status reported to the PICOPROCESSOR by the peripheral device. For example, when the device asserts Function Strobe Acknowledge as a true voltage level to the PICOPROCESSOR, the PICOPROCESSOR inputs a logical 0 to the computer. The following descriptions of the status bits are in terms of the status reported to the processor. The following are the standard status bit assignments for the 32-Bit G-P Intelligent Cable.

Bit 0 - Power. The attached peripheral device asserts Power anytime the peripheral device is attached, powered, online, or some combination of these three. Power On = 0, Power Off = 1.

Bit 1 - Function Strobe Acknowledge (FSA). The attached peripheral device asserts Function Strobe Acknowledge to signal acceptance of data output from the PICOPROCESSOR or the availability of data for input to the computer. FSA = 0, Not FSA = 1.

Bit 2 - Device Busy. The attached peripheral device asserts Device Busy to indicate when it cannot immediately accept data during an output operation or supply data during an input operation. Device Busy = 0, Device Not Busy = 1.

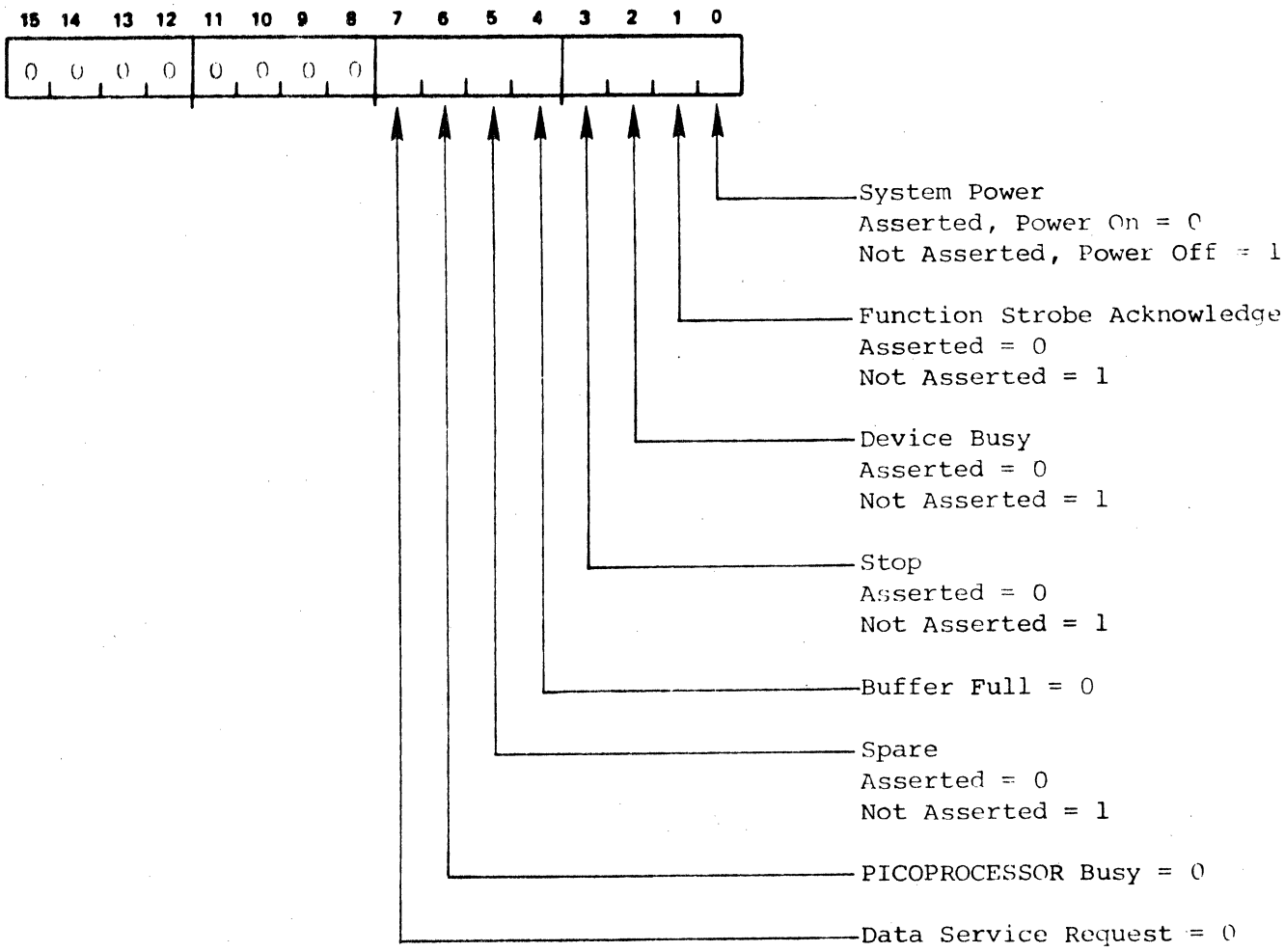
Bit 3 - Stop. The attached peripheral device asserts Stop to indicate terminating conditions were detected in the peripheral device. Normal terminating conditions, such as a filled capacity, or abnormal terminating conditions, such as error conditions, or both can be reported using Stop. Stop = 0, Not Stop = 1.

Bit 4 - Buffer Full. The Buffer Full status bit is derived from PICOPROCESSOR operation and indicates when a full output data word is stored in the PICOPROCESSOR, or when the input data word is fully transferred to the computer. Buffer Full = 0, Buffer Not Full = 1.

Bit 5 - Spare. The Spare is not sampled by the operation sequence. The attached peripheral device can report a device defined bit of status to the computer using Spare. Spare asserted = 0, Spare not asserted = 1.

Bit 6 - PICOPROCESSOR Busy. The PICOPROCESSOR Busy status bit is derived from PICOPROCESSOR operation and must be a logical 1 (not busy) before the PICOPROCESSOR will accept a Branch (begin) command word. PICOPROCESSOR Busy = 0, PICOPROCESSOR Not Busy = 1.

Bit 7 - Data Service Request. The Data Service Request status bit is derived from PICOPROCESSOR operation and indicates when the PICOPROCESSOR is inputting a data service request to the I/O Distributor. Data Service Request = 0, Data Service Not requested = 1.




Any Device Status Lines
Not Connected = 1

Figure 3-88. Status Byte Configuration

3.3.12.3.4 Branch Address Field Description

The 32-Bit G-P Intelligent Cable PICOPROCESSOR uses the standard 4-bit branch address field. The sequence entry point addresses for the 32-Bit G-P Intelligent Cable are expressed at two hexadecimal digits. The least significant digit is provided by the branch address field. The most significant digit is provided by Mode Bit 3, the Input/Output bit.

The allowed entry points to the 32-Bit G-P Intelligent Cable operation sequences are separated into two categories:

- The operation sequences intended for use when the I/O operation is performed using Auto I/O or interrupt I/O programming in the computer.
- The operation sequences intended for use when a non-DMA, non-interrupt (programmed) I/O operation is performed.

The following are the allowed sequence entry point addresses. The normally intended usage for the sequence started at each entry point is included.

:00 - Output Idle. The operation of the 32-Bit G-P Intelligent Cable in programmed I/O mode requires the use of Idle as a sequence entry point. The operation sequence is restored to Output Idle by the computer issuing a branch command word to sequence address :00. Data output using programmed I/O stored in the PICOPROCESSOR with the operation sequence in Output Idle :00.

:01 - Step Pulse. This entry point is used in programmed I/O mode to load Bits 0 and 1 of the mode register into the PICOPROCESSOR's buffer counter and output a Step Pulse to the device followed by Function Strobe.

:02 - Function Strobe. This entry point is used in programmed I/O mode to output Function Strobe to the device.

:03 - Output. This entry point is normally used along with Auto I/O programming in the computer. The Output operation sequence outputs data using automatic implementation of the two wire handshake interface discipline.

:10 - Input Idle. The operation of the 32-Bit G-P Intelligent Cable in programmed I/O mode requires the use of Idle as a sequence entry point. The operation sequence is restored to Input Idle by the computer issuing a Branch command word to sequence address :10. Data input using programmed I/O is performed with the operation sequence in Input Idle :10.

:11 - Step Pulse. This entry point is used in programmed I/O mode to load Bits 0 and 1 of the mode register into the PICOPROCESSOR's buffer counter and output a Step Pulse to the device followed by a return to Input Idle.

:12 - Function Strobe. This entry point is used in programmed I/O mode to output Function Strobe to the device.

:13 - End. This entry point is used in programmed I/O mode to output End to the

3.3.12.4 Operating Sequence

The operation of a peripheral device through the 32-Bit G-P Intelligent Cable is controlled by software in one of the following two ways:

- Auto I/O - If the device interface uses the two wire handshake discipline, then control can be exercised by software direction of the peripheral device with operation sequencing provided by software selection of the appropriate PICOPROCESSOR operation sequence. This method of operation using Auto I/O programming provides software efficiency, both in the amount of software required and in the ability to perform other operations concurrent with the I/O operation.

- Programmed I/O - If the device interface uses an interface discipline other than two wire handshake or if direct software control of the transfer is desired, then control is exercised by software selection of specific parts of the operation sequence with a minimum of sequencing provided internally by the PICOPROCESSOR. This method of operation, called programmed I/O, provides a high degree of operation flexibility but requires constant monitoring of the I/O operation by the computer.

It is possible to use the Input or Output Auto I/O operation sequences along with non-interrupt/non-DMA I/O programming. Under non-interrupt/non-DMA I/O, the software must continuously request status waiting for the data service request status bit 7 = 0. This indicates the 32-Bit G-P Intelligent Cable operation sequence is ready to transfer one byte of data. After the data byte is transferred, the software must simulate acknowledgement of the data service request by software outputting the command word :0200. Since status bit 6, PICOPROCESSOR Busy = 0, the PICOPROCESSOR does not accept the branch address ":0" output as part of the command word.

3.3.12.4.1 Software Sequence (Auto I/O Programming)

All online operations of the peripheral device, when Auto I/O programming is used, are initiated by software issuing a Branch command word to the PICOPROCESSOR. Prior to the issuance of the command, the software must determine the present conditions of the peripheral device. If status was requested and stored at the termination of the previous operation, the software can examine the stored status bits. If an End-of-Block was expected and not received, an operation is in progress. Once the necessary housekeeping is completed, the software can issue the Branch command word.

3.3.12.4.2 Operation Sequences (Auto I/O Programming)

Once initiated, the PICOPROCESSOR's operation sequences assume control of the I/O operation until terminating conditions are sensed. The following describes the operation sequences flow charted in Figures 3-89 and 3-90.

IDLE (:00 or :10)

The PICOPROCESSOR normally loops in Idle whenever the input or output sequence is not being performed. The Branch command word used to initiate either the output or input sequence is normally issued when the operation sequence is in Idle.

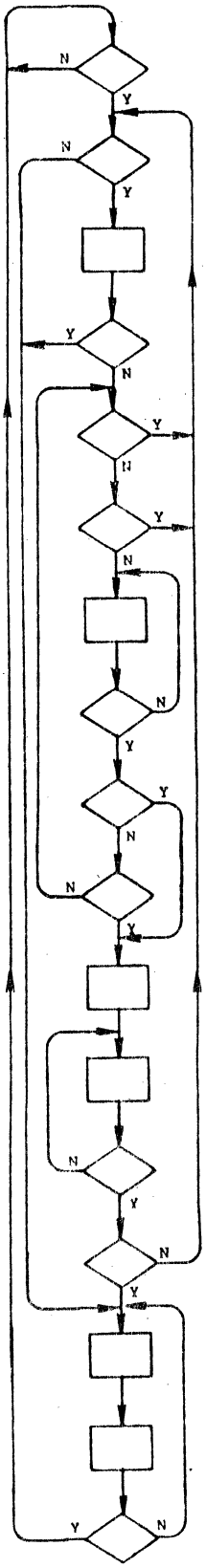
**NOTE**

The PICOPROCESSOR will accept a Branch command word anytime it is reporting PICOPROCESSOR Not Busy status. This feature is used mainly with programmed I/O.

OUTPUT SEQUENCE (:03)

The output sequence starts with a system check for the device inputting Power On status. Failure of the system check results in the operation being aborted. If the device is powered, the configuration of Mode Bits 0 and 1 is loaded into the buffer counter and then a check for the device inputting Stop status is made. As with the system check, failure results in the operation being aborted. When the operation is aborted, the PICOPROCESSOR issues End to the device and an End-of-Block service request to the I/O Distributor to indicate terminating conditions were detected. The End-of-Block service request is held until acknowledged, at which time the operation sequence returns to Idle.

Normally the operation does not abort and the output sequence continues by checking for the device inputting Busy status or a Function Strobe Acknowledge. If either status is present, the sequence keeps looping back to the start of the output sequence until the device is fully ready to receive data. Once the device is ready, the PICOPROCESSOR issues a data service request to the I/O Distributor. When the request is acknowledged, the data byte output by the I/O Distributor is loaded into the appropriate location in the PICOPROCESSOR's buffer register. After the data byte is stored, the Auto I/O byte count is checked for zero and the sequence skips the Buffer Full check if the byte count is zero. If the byte count is not zero, the next check is for the Buffer Full signal which indicates a full output word is stored in the PICOPROCESSOR. If a full output word has not been stored, the buffer counter is incremented and the sequence loops back to the device checks to request another byte of data from the I/O Distributor. This loop is repeated until the Buffer Full signal or an Auto I/O byte count of zero is detected. The PICOPROCESSOR then issues Step to the device followed by Function Strobe. The Function Strobe is held until the device responds with a Function Strobe acknowledge. Once Function Strobe Acknowledge is received, the sequence loops back to the start of the Output sequence if more data must be transferred. When the Auto I/O byte count does go to zero, the PICOPROCESSOR issues End to the device and an End-of-Block service request to the I/O Distributor to indicate terminating conditions were detected. The End-of-Block service request is held until acknowledged, at which time the operation sequence returns to Idle.




Idle :00
Was a Branch command received?

Entry Point :03
Is the device inputting Power On status?

Load mode bits 0 and 1 into the buffer counter.

Is the device inputting Stop status?

Is the device inputting Busy status?

Is the device inputting Function Strobe Acknowledge?

Issue a data service request to the I/O Distributor.

Was the request acknowledged?

Did the Auto I/O byte count go to zero with this data transfer?

Is the buffer counter signalling Buffer Full?

Issue Step to the device.

Issue Function Strobe to the device.

Is the device inputting Function Strobe Acknowledge?

Did the Auto I/O byte count go to zero with this data transfer?

Issue End to the device.

Issue an End-of-Block service request to the I/O Distributor.

Was the request acknowledged?

Figure 3-89. Output Sequence 32-Bit G-P Intelligent Cable

INPUT SEQUENCE (:14)

The input sequence starts with Step being issued to the device to signal that the PICOPROCESSOR is ready to accept a data word from the device. A check is then made for the device inputting Stop status. Failure of this check results in the operation being aborted. When the operation is aborted, the PICOPROCESSOR issues End to the device and an End-of-Block service request to the I/O distributor to indicate terminating conditions were detected. The End-of-Block service request is held until acknowledged, at which time the operation sequence returns to Idle.

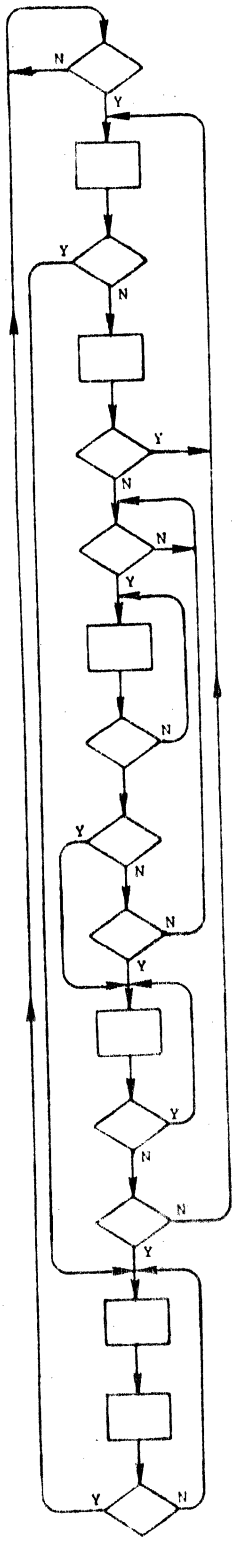
If the device is not inputting Stop status, the configuration of mode bits 0 and 1 is loaded into the buffer counter and then a check is made for the device inputting Busy status. If the device is inputting Busy status, the sequence keeps looping back to the start of the input sequence until the device is fully ready to input data. The sequence then checks for a Function Strobe Acknowledge from the device to signal that a data word is being presented for input. The sequence waits at this point until Function Strobe Acknowledge is received.

**NOTE**

The device must present the data word and Function Strobe Acknowledge until the PICOPROCESSOR issue Function Strobe to the device.

When the Function Strobe Acknowledge is received, the PICOPROCESSOR issues a data service request to the I/O distributor. When the request is acknowledged, the appropriate data byte is input to the computer. After the data byte is input, the check for an Auto I/O byte count of zero is made and the Buffer Full check if the byte count is zero. If the byte count is not zero, the next check is for the Buffer Full signal which indicates the full data word has been input to the computer. If a full data word has not been input, the buffer counter is incremented and the sequence loops back to the Function Strobe Acknowledge check to request input of another byte of data. This loop is repeated until the buffer full signal or an Auto I/O byte count of zero is detected. The PICOPROCESSOR then issues Function Strobe to the device to signal acceptance of the input data word.

The Function Strobe is held until the device drops Function Strobe Acknowledge. The sequence loops back to the start of the input sequence if more data must be transferred. When the Auto I/O byte count does go to zero, the PICOPROCESSOR issues End to the device and an End-of-Block service request to the I/O distributor to indicate terminating conditions were detected. The End-of-Block service request is held until acknowledged, at which time the operation sequence returns to Idle.




Idle :10
Was A Branch command received?

ENTRY POINT :14
Issue Step to the device.

Is the device inputting Stop status?

Load mode bits 0 and 1 into the buffer counter.

Is the device inputting Busy status?

Is the device inputting Function Strobe Acknowledge?

Issue a data service request to the I/O distributor.

Was the request acknowledged?

Did the byte count go to zero with this data transfer?

Is the byte counter signalling Buffer Full?

Issue Function Strobe to the device.

Is the device inputting Function Strobe Acknowledge?

Did the byte count go to zero with this data transfer?

Issue End to the device.

Issue an End-of-Block service request to the I/O distributor.

Was the request acknowledged?

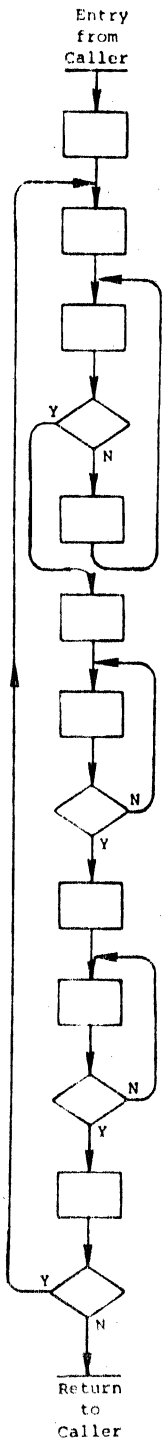
Figure 3-90. Input Sequence 32-Bit G-P Intelligent Cable

3.3.12.4.3 Software Sequence (Programmed I/O)

The form of any software sequence for programmed I/O with the 32-Bit G-P Intelligent Cable is dependent on the attached device, its application, and its interface discipline requirements (refer to Section 3.3.12.6, applications and programming examples). Due to the large total number of possible programmed I/O sequences, description of any comprehensive selection of programmed I/O sequences is beyond the scope of this document. To give an insight into the programming methods used to implement an interface discipline in programmed I/O, flow charts and descriptions of programmed I/O sequences corresponding to the two wire handshake interface discipline are provided in this section. Both sequences are shown as subroutines called by the mainline program.

OUTPUT

The output programmed I/O sequence (Figure 3-91) starts with software issuing a Reset command word to clear and initialize the logic of the PICOPROCESSOR. Software then issues a Set Mode command word to select output mode and specify the output word size. This also loads the buffer counter. Software then requests status from the PICOPROCESSOR and checks for Buffer Full status. If the PICOPROCESSOR reports Buffer Not Full status, software outputs one byte of data and again requests status from the PICOPROCESSOR. When the PICOPROCESSOR reports Buffer Full status, software outputs one additional byte of data, requests status from the PICOPROCESSOR and checks the status for the device being ready to receive data. Device ready status consists of Function Strobe Acknowledge not asserted and the appropriate state for any other system status bits input by the device. If the device is not ready, software continues to request and check status until the device is ready. Software then issues a Branch command word to firmware entry point :01 to assert Function Strobe and Step to the device. Software then requests status from the PICOPROCESSOR and checks for the device asserting Function Strobe Acknowledge. If the device is not asserting Function Strobe Acknowledge, software continues requesting and checking status until the device responds with Function Strobe Acknowledge. Software then issues a Branch command word to firmware entry point :00 to drop Function Strobe. If more data transfers are required, software returns to the Set Mode step to output the next data word. When all data transfers are completed, software returns to the caller in the mainline program.




Issue a Reset command word to initialize the PICOPROCESSOR.

Issue a Set Mode command word to select output and to load the buffer counter.

Request status from the PICOPROCESSOR.

Did the PICOPROCESSOR input Buffer Full status?

Output one byte of data.

Output the last byte of data.

Request status from the PICOPROCESSOR.

Is the device ready to accept data?

Issue a Branch command word to firmware entry point :01 to assert Function Strobe and pulse Step to the device.

Request status from the PICOPROCESSOR.

Is the device inputting Function Strobe Acknowledge?

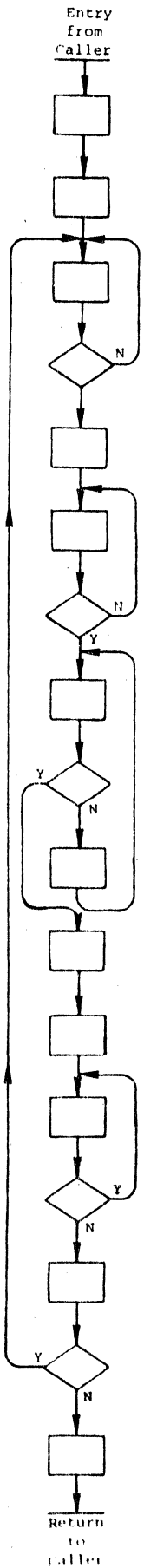
Issue a Branch command word to firmware entry point :00 to drop Function Strobe.

Are more data transfers required?

Figure 3-91. Output Programmed I/O Sequence

INPUT

The input programmed I/O sequence (Figure 3-92) starts with software issuing a Reset command word to clear and initialize the logic of the PICOPROCESSOR. Software then issues a Set Mode command word to select input mode and specify the input word size. Software then requests status from the PICOPROCESSOR and checks for the device ready to input data. Device ready consists of the appropriate state of the system status bits input by the device. If the device is not ready, software continues to request and check status until the device is ready. When the device signals ready, software issues a Branch command word to firmware entry point :11 to pulse Step and load the buffer counter. Software again requests status and checks for the device asserting Function Strobe Acknowledge. If the device is not asserting Function Strobe Acknowledge, software continues requesting and checking status until the device responds with Function Strobe Acknowledge. Software then requests status and checks for Buffer Full status. If the PICOPROCESSOR reports Buffer Not Full status, software inputs one byte of data and again requests status from the PICOPROCESSOR. This continues until the PICOPROCESSOR reports Buffer Full status. Software then inputs one additional byte of data followed by a Branch command word to firmware entry point :12 to assert Function Strobe to the device to signal acceptance of the data word. Software then requests status and check for the device asserting Function Strobe Acknowledge. If the device is asserting Function Strobe Acknowledge, software continues requesting and checking status until the device drops Function Strobe Acknowledge. Software then issues a Branch command word to firmware entry point :10 to drop Function Strobe to the device. If more data transfers are required, software returns to the status request to check device ready in preparation for inputting the next data word. When all data transfers are completed, software returns to the caller in the mainline program.




Issue a Reset command word to initialize the PICOPROCESSOR.

Issue a Set Mode command word to select input and to load the buffer counter.

Request status from the PICOPROCESSOR.

Is the device ready to input data?

Issue a Branch command word to firmware entry point :11 to pulse step and load the buffer counter.

Request status from the PICOPROCESSOR.

Is the device inputting Function Strobe Acknowledge?

Request status from the PICOPROCESSOR.

Did the device input Buffer Full status?

Input one byte of data.

Input the last byte of data.

Issue a Branch command word to firmware entry point :12 to assert Function Strobe to the device.

Request status from the PICOPROCESSOR.

Is the device inputting Function Strobe Acknowledge?

Issue a Branch command word to firmware entry point :10 to drop Function Strobe.

Are more data transfers required?

Issue a Branch command word to firmware entry point :13 to assert End to the device.

Figure 3-92. Input Programmed I/O Sequence

3.3.12.5 Interface Description

The 32-Bit G-P Intelligent Cable has two interfaces:

I/O Distributor to PICOPROCESSOR

PICOPROCESSOR to device

The I/O Distributor to PICOPROCESSOR interface is the standard Intelligent Cable interface.

The PICOPROCESSOR to device interface is carried over two flat ribbon cables. The data lines are carried over a 48-conductor cable and the control lines are carried over a separate 16-conductor cable. The device cables are not supplied with a device mating connector. Figure 3-93 is the PICOPROCESSOR to device interface overview. Table 3-16 lists the signal assignments for the 48-conductor cable and Table 3-17 lists the signal assignments for the 16-conductor cable.

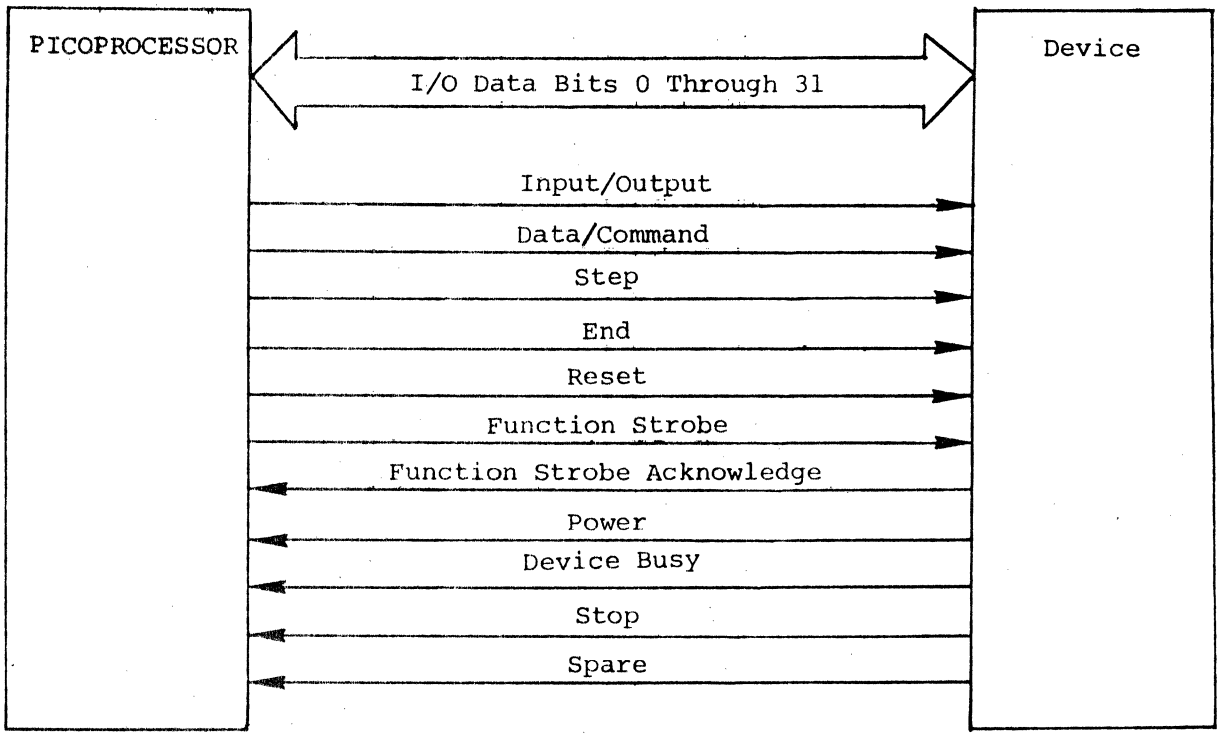



Figure 3-93. Interface Overview

Table 3-16. Data Cable Interface Signal Assignment

| PICOPROCESSOR Connector | Pin | Description | Device Cable Wire Number (Color) |
|---|---|---|---|
| P4 | 8 | Data Bit 1 | 1 (Brown) |
| P4 | 9 | Data Bit 5 | 2 (Red) |
| P4 | 7 | Ground | 3 (Orange) |
| P4 | 10 | Ground | 4 (Yellow) |
| P4 | 6 | Data Bit 6 | 5 (Green) |
| P4 | 11 | Data Bit 0 | 6 (Blue) |
| P4 | 5 | Data Bit 3 | 7 (Violet) |
| P4 | 12 | Data Bit 2 | 8 (Gray) |
| P4 | 4 | Ground | 9 (White) |
| P4 | 13 | Ground | 10 (Black) |
| P4 | 3 | Data Bit 8 | 11 (Brown) |
| P4 | 14 | Data Bit 9 | 12 (Red) |
| P4 | 2 | Data Bit 7 | 13 (Orange) |
| P4 | 15 | Data Bit 4 | 14 (Yellow) |
| P4 | 1 | Ground | 15 (Green) |
| P4 | 16 | Ground | 16 (Blue) |
| P5 | 8 | Data Bit 11 | 17 (Violet) |
| P5 | 9 | Data Bit 10 | 18 (Gray) |
| P5 | 7 | Ground | 19 (White) |
| P5 | 10 | Ground | 20 (Black) |
| P5 | 6 | Data Bit 18 | 21 (Brown) |
| P5 | 11 | Data Bit 13 | 22 (Red) |
| P5 | 5 | Data Bit 19 | 23 (Orange) |
| P5 | 12 | Data Bit 14 | 24 (Yellow) |
| P5 | 4 | Ground | 25 (Green) |
| P5 | 13 | Ground | 26 (Blue) |
| P5 | 3 | Data Bit 17 | 27 (Violet) |
| P5 | 14 | Data Bit 15 | 28 (Gray) |
| P5 | 2 | Data Bit 16 | 29 (White) |
| P5 | 15 | Data Bit 12 | 30 (Black) |
| P5 | 1 | Ground | 31 (Brown) |
| P5 | 16 | Ground | 32 (Red) |
| P6 | 8 | Data Bit 23 | 33 (Orange) |
| P6 | 9 | Data Bit 21 | 34 (Yellow) |
| P6 | 7 | Data Bit 22 | 35 (Green) |
| P6 | 10 | Data Bit 20 | 36 (Blue) |
| P6 | 6 | Ground | 37 (Violet) |
| P6 | 11 | Ground | 38 (Gray) |
| P6 | 5 | Data Bit 29 | 39 (White) |
| P6 | 12 | Data Bit 25 | 40 (Black) |
| P6 | 4 | Data Bit 28 | 41 (Brown) |
| P6 | 13 | Data Bit 24 | 42 (Red) |
| P6 | 3 | Ground | 43 (Orange) |
| P6 | 14 | Ground | 44 (Yellow) |
| P6 | 2 | Data Bit 30 | 45 (Green) |
| P6 | 15 | Data Bit 27 | 46 (Blue) |
| P6 | 1 | Data Bit 31 | 47 (Violet) |
| P6 | 16 | Data Bit 26 | 48 (Gray) |

Table 3-17. Control Cable Interface Signal Assignments

| PICOPROCESSOR Connector | Pin | Description | Device Cable Wire Number (Color) |
|---|---|---|---|
| P3 | 8 | Stop | 1 (Brown) |
| P3 | 9 | Ground | 2 (Red) |
| P3 | 7 | Ground | 3 (Orange) |
| P3 | 10 | Function Strobe Acknowledge | 4 (Yellow) |
| P3 | 6 | Spare | 5 (Green) |
| P3 | 11 | Reset | 6 (Blue) |
| P3 | 5 | Ground | 7 (Violet) |
| P3 | 12 | Step | 8 (Gray) |
| P3 | 4 | Power | 9 (White) |
| P3 | 13 | End | 10 (Black) |
| P3 | 3 | Ground | 11 (Brown) |
| P3 | 14 | Input/Output | 12 (Red) |
| P3 | 2 | Ground | 13 (Orange) |
| P3 | 15 | Data/Command | 14 (Yellow) |
| P3 | 1 | Busy | 15 (Green) |
| P3 | 16 | Function Strobe | 16 (Blue) |

3.3.12.5.1 Device Interface Line Description

The signal interface lines between the PICOPROCESSOR and the device consist of 32 bidirectional data lines, six output control lines, and 5 input control/status lines. The PICOPROCESSOR to device interface is a negative true interface, that is, a logical 1 = 0V (nominal) and a logical 0 = +5V (nominal).

DATA LINES

The 32-Bit G-P Intelligent Cable has 32 parallel, bidirectional, data lines numbered 0 through 31 with line 31 being the most significant bit. Any number or configuration of the data lines may be used by the device. Unused lines need not be terminated. All ground lines in the 48-conductor cable should be connected to device logic ground to provide adequate ground return for the signal lines.

The 32-Bit G-P Intelligent Cable provides a 32-bit output storage buffer with TTL tri-state output drivers for the 32 data lines. A corresponding set of 32 TTL tri-state input receivers are used for data input. The data line drivers are capable of sinking 32 mA and the data line receivers present one TTL load. All data lines are pulled up in the PICOPROCESSOR to +5 volts through 10K Ohm resistors. The length of the 48-conductor data cable should be limited to the 0.46 m (1.5 ft.) provided. Refer to Figure 3-94. The device portion of the interface assumes an input/output device and shows both drivers and receivers. As an alternate to the interface shown, tri-state transceivers could be used in the device. For a simplex device, the unused drivers or receivers are not required.

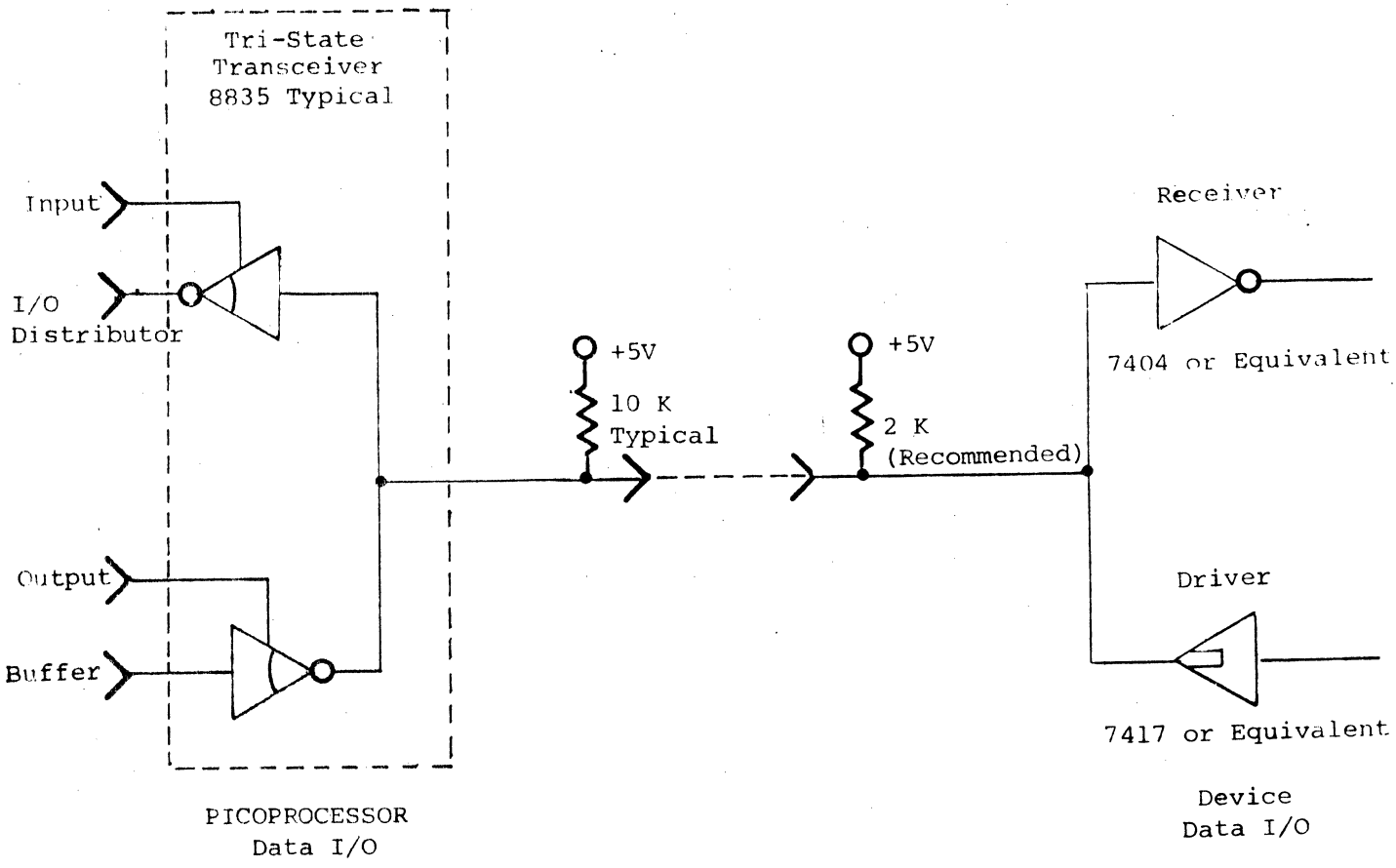



Figure 3-94. PICOPROCESSOR Data Lines Logic

A storage buffer is not provided for the input data word. The loading of the output storage buffer or the acceptance of an input data word is performed in 8-bit bytes starting from the most significant bit.

Data transferred as 32-bit words uses the full buffer capacity of the 32-Bit G-P Intelligent Cable. Data transferred as 8-, 16-, or 24-bit words use a corresponding smaller amount of the buffer capacity. Specifically, an 8-bit word is transferred over data lines 0 through 7; a 16-bit word is transferred over data lines 0 through 15; and a 24-bit word is transferred over data lines 0 through 23.

OUTPUT CONTROL LINES

The use of the Output control lines by the device is dependent on the type of operation being performed by the device. For example, if the 32-Bit G-P Intelligent Cable is used to drive a static display panel or similar unsequenced types of operations, none of the control lines need be used. When sequencing of the I/O Transfer is required, then one or more of the control lines must be used. The Output control lines are driven by open collector drivers capable of sinking 32 mA with a pull-up to +5 Vdc provided on the PICOPROCESSOR. Refer to Figure 3-95. Unused control lines need not be terminated. All ground lines in the 16-conductor cable should be connected to device logic ground to provide adequate ground return for the signal lines. The following describes the functions associated with each of the control lines.

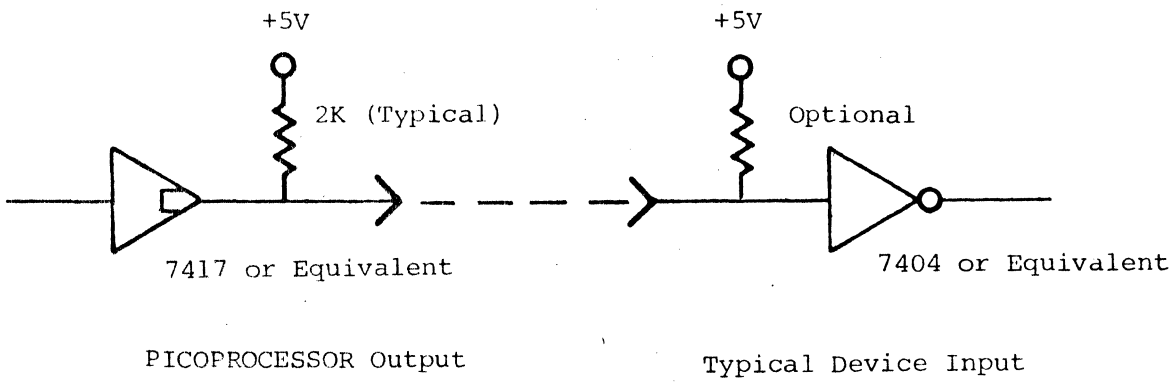



Figure 3-95. PICOPROCESSOR Output Control Logic

INPUT/OUTPUT. The Input/Output control line is driven by the Input/Output bit in the mode register. This line is a logical 0 for output operations and a logical 1 for input operations. Software determines the state of the Input/Output control line by issuing the appropriate Set Mode command word or by issuing a Reset command word. Following the Reset command word, the state of the Input/Output control line is logical 0.

Use of the Input/Output control line by the device is optional and it is normally used by devices capable of both accepting or providing data.

DATA/COMMAND. The Data/Command line is driven by the Data/Command bit in the mode register. This line is a logical 0 for command operations and a logical 1 for data operations. Software determines the state of the Input/Output control line by issuing the appropriate Set Mode command word or by issuing a Reset command word. Following the Reset command word, the state of the Data/Command control line is logical 0.

Use of the Data/Command line by the device is optional. Typically this line is used by devices that can accept both remote programming and data or devices that can provide both external (to the PICOPROCESSOR operation) status and data. This line is unused in the PICOPROCESSOR and may be redefined to suit the needs of the device.

STEP. The Step control line is driven by a 500 ns (nominal) single shot and is derived from the operation sequences. The resulting step pulse is a 500 ns logical 1 pulse. The Step pulse is generated automatically by the operation sequences when Automatic I/O programming is used and can be generated during programmed I/O by software issuing a Branch command word to the appropriate sequence entry point.

As used during the output sequence, the Step pulse signals the device that the data available is a valid data word. As used during the input sequence, the Step pulse signals that the PICOPROCESSOR is ready to accept data from the device. Use of the Step control line by the device is optional and is normally used by devices that require a strobed type of interface discipline.

END. The End control line is derived from the operation sequences and is asserted as a logical 1. The End control line is asserted automatically by the operation sequences when Automatic I/O programming is used and can be generated during programmed I/O by software issuing a Branch command word to the appropriate sequence entry point. As used in both the input and output sequences, the End control line is asserted to signal the device that terminating conditions were encountered.

Use of the End control line by the device is optional and its use is not related to any particular type of operation.

RESET. The Reset control line passes on the Reset signal used to initialize the PICOPROCESSOR. The Reset control line is asserted as a logical 1. The Reset control line is asserted anytime software issues a Reset command word or whenever general Reset occurs on the Maxi-Bus.

Use of the Reset control line by the device is optional and its use is not related to any particular type of operation.

FUNCTION STROBE. The Function Strobe control line is derived from the operation sequences and is asserted as a logical 1. The Function Strobe control line is asserted automatically by the operation sequences as part of the two wire handshake when Automatic I/O programming is used. Function Strobe can be generated during programmed I/O by software issuing a Branch command word to the appropriate sequence entry point.

Use of the Function Strobe control line by the device is mandatory if Automatic I/O programming and the operation sequences are used to perform the data tranfers. Its use with programmed I/O is optional.

INPUT CONTROL/STATUS LINES

The use of the Input Control/Status lines by the device is dependent on the type of operation being performed by the device. For example, if the 32-Bit C-P Intelligent Cable is used to poll the switch settings on a control panel or similar unsequenced types of operations, none of the control lines need be used. When sequencing of the I/O Transfer is required, then one or more of the control lines must be used. The input receivers for the Control/Status lines present one TTL load plus a pull-up to +5 volts through 2K ohms resister. Refer to Figure 3-96. Unused control lines should be attached as noted in the descriptions. All ground lines in the 16-conductor cable should be connected to device logic ground to provide adequate ground return for the signal lines used. The following describes the functions associated with each of the control lines.

**NOTE**

A logic inversion occurs in the PICOPROCESSOR between the Control/Status lines as input to the PICOPROCESSOR and the status as reported to the computer. Thus in terms of the PICOPROCESSOR Control/Status lines (negative true), a line is asserted a logical 1 (ground) and not asserted as a logical 0 (+5V). In terms of the status stored when software requests status, a line is asserted as a logical 0 and not asserted as a logical 1.

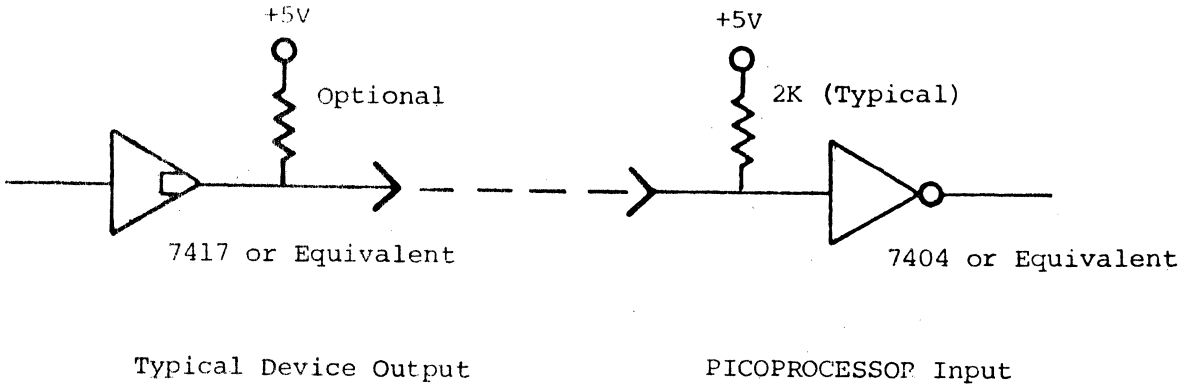



Figure 3-96. PICOPROCESSOR Input Control/Status Logic

FUNCTION STROBE ACKNOWLEDGE. The Function Strobe Acknowledge control/status line is sampled automatically by the input and output sequences as part of the two wire handshake. The Function Strobe Acknowledge control/status line can be sampled during programmed I/O by software requesting status from the PICOPROCESSOR.

Use of the Function Strobe Acknowledge control/status line by the device is mandatory if Automatic I/O programming and the operation sequences are used to perform the data transfers. Its use with programmed I/O is optional.

POWER. The Power control/status line is sampled automatically by the input and output sequences as a system check to verify the associated condition in the device. The Power control/status line can be sampled during programmed I/O by software requesting status from the PICOPROCESSOR.

Use of and the specific definition for the Power control/status line by the device is optional. If Automatic I/O programming and the operation sequences are used to perform the data transfers, a logical 1 (device ground) signal must be provided on this line to assure proper operation sequencing.

DEVICE BUSY. The Device Busy control/status line is sampled automatically by the input and output sequences as a system check to verify the associated condition in the device. The Busy control/status line can be sampled during programmed I/O by software requesting status from the PICOPROCESSOR.

Use of and the specific definition for the Busy control/status line by the device is optional. If Automatic I/O programming and the operation sequences are used to perform the data transfers, a logical o signal must be provided on this line (or the line left unconnected) to assure proper sequencing of the firmware.

STOP. The Stop control/status line is sampled automatically by the input and output sequences to check for the device signaling that terminating conditions were encountered. The Stop control/status line can be sampled during programmed I/O by software requesting status from the PICOPROCESSOR.

Use of and the specific definition for the Stop control/status line by the device is optional. If Automatic I/O programming and the operation sequences are used to perform the data transfers, a logical 0 signal must be provided on this line (or the line left unconnected) to assure proper sequencing of the firmware.

SPARE. The Spare control/status line is not sampled by the input and output sequences. The Spare control/status line can be sampled by software requesting status from the PICOPROCESSOR.

Use of and the specific definition of the Spare control/status line by the device is optional. There are no connection requirements for the Spare control/status line.

3.3.12.5.2 Interface Timing Requirements

The timing requirements presented here apply specifically when the input and output sequences are used along with Automatic I/O programming to control the transfer of data. If the peripheral device is unable to observe the timing requirements shown here, the computer software should be designed to operate the PICOPROCESSOR, under Programmed I/O, in a manner that observes the device's interface discipline. For unsequenced operations, such as polling the attached peripheral device's control settings, the timing requirements are eliminated.

The timing diagrams shown in Figures 3-97 and 3-98 use the following conventions:

- Sequence "bubbles" not separated by a time break (—≈—) are sequenced at intervals equal to or greater than 250 ns.
- The time breaks indicate indeterminate duration computer or peripheral device operations.
- The crosshatched sections of the timing diagrams indicate either a don't know or a don't care situation. See the text for additional information.
- The various signals are shown in the negative true form used on the PICOPROCESSOR to device interface.

OUTPUT TIMING

The timing diagram shown in Figure 3-97 is separated into three distinct zones. The first zone, initialization shows the effect of a Reset command word (or general Maxi-Bus reset) on the 32-Bit G-P Intelligent Cable. The control/status signals input from the device may or may not be reset during initialization.

The second zone, data transfers, consists of a closed timing loop started by software issuing a Branch command word and terminated by either the computer or the device. Power and Stop are shown normalized prior to the issuance of the command word since an improper level detected on either of these lines would abort the operation. Since Device Busy and Function Strobe Acknowledge both provide wait loops if the improper level is detected, they are shown as occurring after the issuance of the command word. They also can be used to delay subsequent data transfers until the device is ready to accept more data. When Function Strobe Acknowledge is checked successfully, the PICOPROCESSOR proceeds to fill the output buffer register. Since up to four data transfers may be required to fill the buffer, data is shown on the data lines but it is not stable or valid until just before the Step pulse is issued. Step is then followed by Function Strobe which is held indefinitely until the device responds with Function Strobe Acknowledge. Device Busy is shown as occurring in response to the Step pulse. This is the first point at which the device can safely go Busy. Following receipt of Function Strobe Acknowledge, the PICOPROCESSOR drops Function Strobe. If more data must be transferred, the sequence loops back to the Power check. If this was the last data transfer, the sequence proceeds to the termination phase. If the device wishes to terminate the operation, it must assert Stop which causes the operation to abort on the next pass through the sequence. Stop is shown timed to the leading edge of Function Strobe Acknowledge to assure detection on the next pass through the sequence.

The termination zone shows End being asserted while the End-of-Block operation is

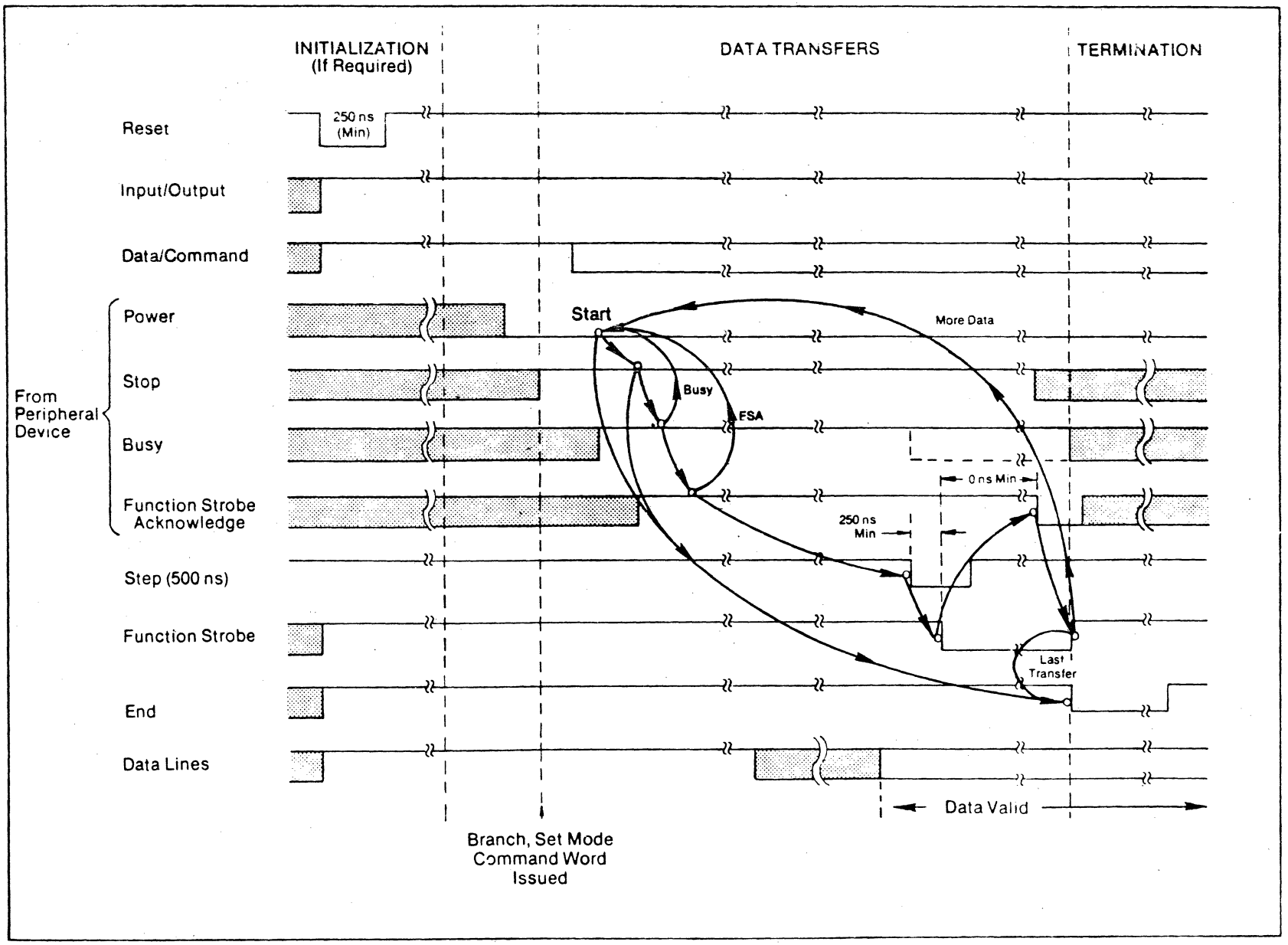



Figure 3-97. Output Firmware Sequence Timing

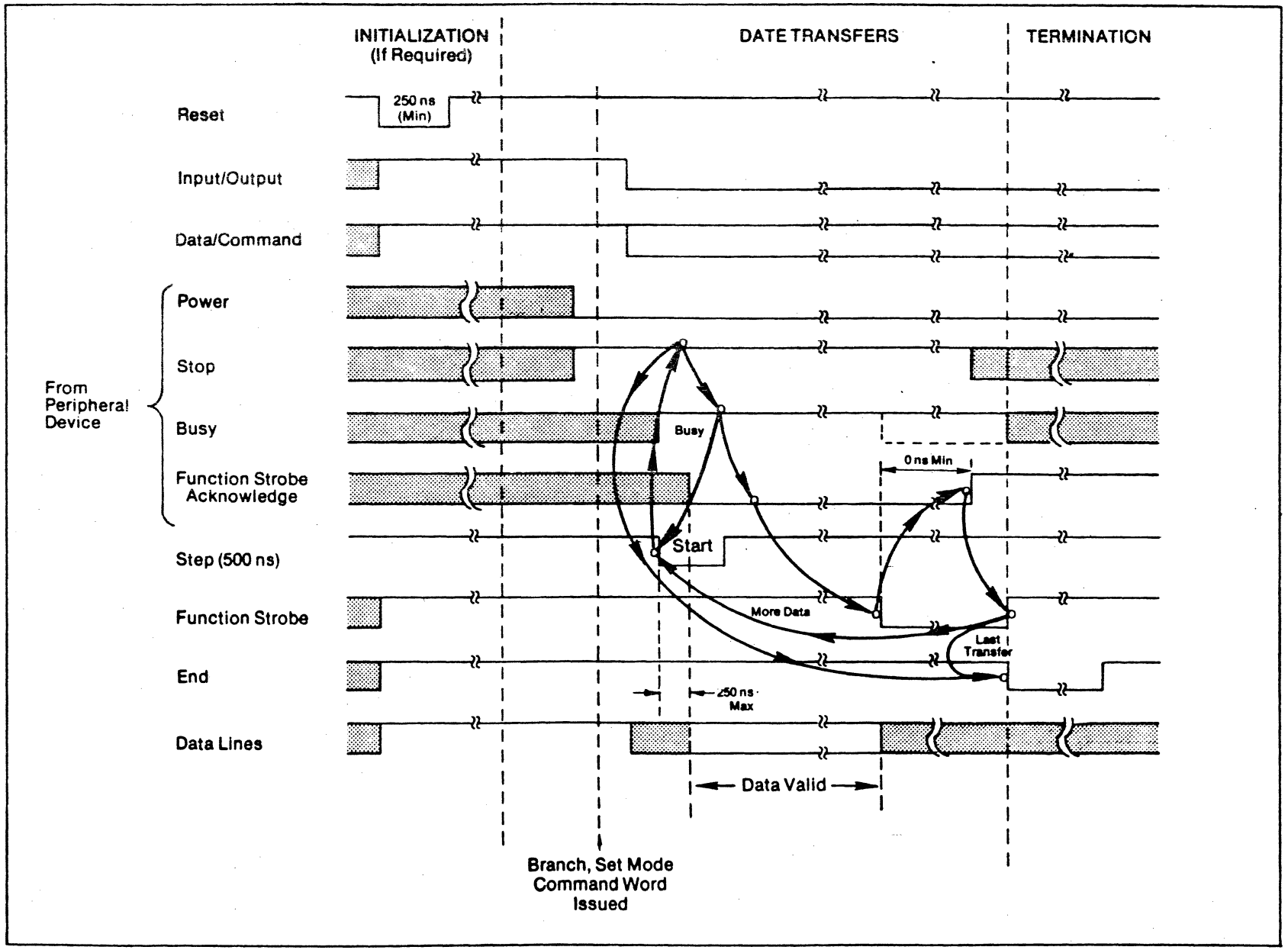



Figure 3-98. Input Firmware Sequence Timing

INPUT TIMING

The timing diagram shown in Figure 3-98 is separated into three distinct zones. The first zone, initialization shows the effect of a Reset command word (or general Maxi-Bus reset) on the 32-Bit G-P Intelligent cable. The control/status signals input from the device may or may not be reset during initialization. A specific point concerning initialization, the PICOPROCESSOR is enabled for output by Reset and the data lines are being driven as all zeros.

The second zone, data transfers, consists of a closed timing loop started by software issuing a command word and terminated by either the computer or the device. Power and Stop are shown normalized prior to the issuance of the command word. Power is not checked by the input firmware sequence; however, Stop is and an improper level detected on this line would abort the operation. Following the issuance of the command word, the PICOPROCESSOR switches from output to input. At this point, the PICOPROCESSOR stops driving the data lines and the input data is present on the data lines. The Step line is pulsed, then Stop is checked, followed by Device Busy being checked. If the device signals Device Busy, the Step pulse is re-triggered and the Stop and Busy checks repeated. When Busy is checked successfully, the PICOPROCESSOR checks for Function Strobe Acknowledge and waits indefinitely until it receives Function Strobe Acknowledge. *During input, the device must present valid data within 250 ns of the presenting of Function Strobe Acknowledge and must hold both the data and Function Strobe Acknowledge stable until Function Strobe is asserted by the PICOPROCESSOR.* Once asserted, the PICOPROCESSOR holds Function Strobe asserted indefinitely until the device responds by dropping Function Strobe Acknowledge. Device Busy is shown occurring in response to the PICOPROCESSOR asserting Function Strobe. This is the first point at which the device can safely assert Device Busy. Following receipt of Function Strobe Acknowledge, the PICOPROCESSOR drops Function Strobe. If more data must be transferred, the sequence loops back to pulse Step. If this was the last data transfer, the sequence proceeds to the termination phase. If the device wishes to terminate the operation, it must assert Stop which causes the operation to abort on the next pass through the sequence. Stop is shown timed to the trailing edge of Function Strobe Acknowledge to assure detection on the next pass through the sequence.

The termination zone shows End being asserted while the End-of-Block operation is being completed.

### 3.3.12.6 Applications and Programming Examples

The applications presented in this section give insight into the methods used to implement an interface using the 32-Bit G-P Intelligent Cable. The programming examples illustrate how software is then designed to work with the interface.

#### 3.3.12.6.1 Computer to Computer Link

The computer to computer link presented here is a possible application example designed to connect two LSI Family computers together using two cross coupled 32-Bit G-P Intelligent Cables.

The basic computer to computer link is shown in Figure 3-99. The minimum control line connections for this type of operation are Function Strobe cross connected with Function Strobe Acknowledge and Power connected to ground. Successful operation is possible with this minimum control connection. To allow either computer to terminate the data transmission, End is cross connected with Stop. This assures that successful completion of the data transmission by one computer will force an orderly operation termination in the other computer. To provide a means for the two Computers to establish a data transmission link in an orderly manner, Request Service (redefinition of Data/Command) is cross connected with Acknowledge (redefinition of Spare). Data/Command and Spare are not used internally by the PICOPROCESSOR so the operation sequences are not affected by the redefinition; however, a software protocol is required to determine which computer becomes the sender and which computer is the receiver.

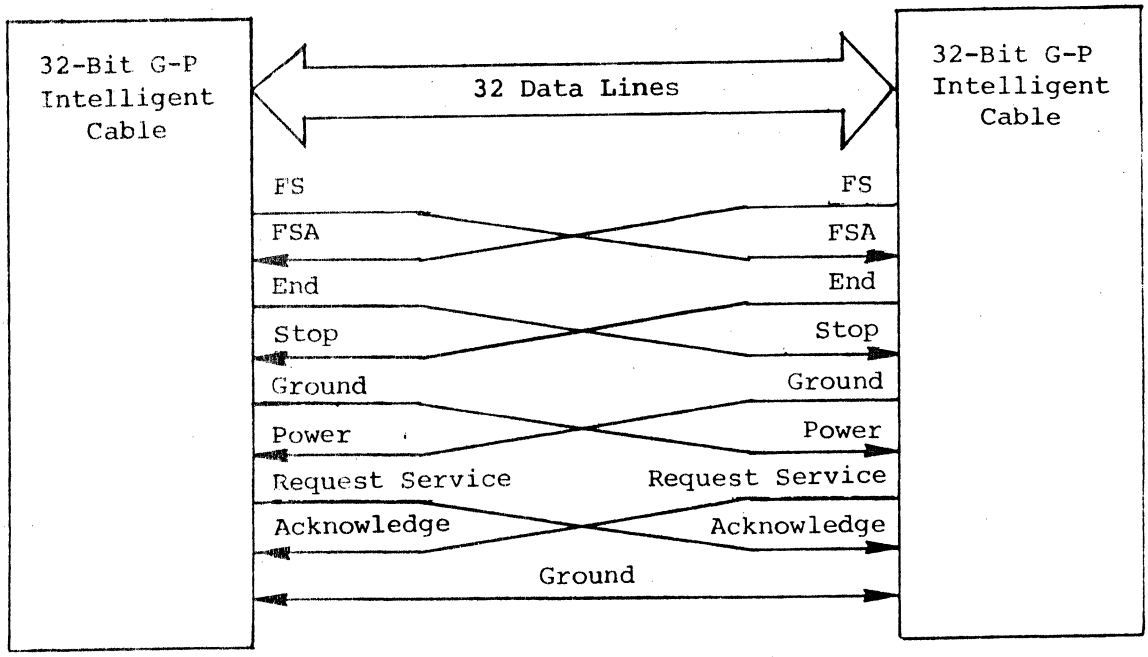



Figure 3-99. Computer to Computer Link Overview

Programming Example

The Assembler Language statements shown in the following example demonstrate a basic program for using the computer to computer link application. This example shows the various routines necessary to initiate a data transmission. The demonstration code would be used in both computers.

The first section of code is the parameter equates that define a series of constants used to assemble the various routines. The first five establish the device address and interrupt vector locations. The remaining equates select Intelligent Cable operations.

```
*
* PARAMETER EQUATES FOR 32-BIT G-P INTELLIGENT CABLE
*
INTBAS    EQU       :C0           INTERRUPT VECTOR BASE (STRAPPED ON I/O DISTRIBUTOR)
CHAN      EQU       (specify)     INTELLIGENT CABLE IS PLUGGED INTO THIS CHANNEL
IOD       EQU       :F            ADDRESS BASE (STRAPPED ON I/O DISTRIBUTOR
INTAD     EQU       CHAN%3+INTBAS DATA INTERRUPT VECTOR ADDRESS FOR INTELLIGENT CABLE
GPDA      EQU       IOD%3+CHAN%1  INTELLIGENT CABLE DEVICE ADDRESS
*
SETMOD    EQU       :400          SET MODE COMMAND WORD BASE
BYTE 2    EQU       :1            SELECT 16-BIT WORD SIZE
BYTE 3    EQU       :2            SELECT 24-BIT WORD SIZE
BYTE 4    EQU       :3            SELECT 32-BIT WORD SIZE
WORD      EQU       BYTE 4        SELECTS THE WORD SIZE USED
RQST      EQU       :4            REQUEST SERVICE SIGNAL
BEGIN     EQU       :200          BRANCH COMMAND WORD BASE
OUT       EQU       :30           OUTPUT
IN        EQU       :48           INPUT SEQUENCE ADDRESS
```

The following sections of code, interrupt vectored locations (output) and interrupt vectored locations (input), defines and fills the memory locations associated with the I/O Distributor channel to which the 32-Bit G-P Intelligent Cable is connected. There are six memory locations involved with each of these sections of code; however, these are the same six memory locations for both sections of code since all data service and End-of-Block service interrupts for the I/O Distributor channel where the 32-Bit G-P Intelligent Cable is connected are vectored to the same memory addresses. Each time the Automatic I/O operation must be switched from input to output or output to input, the interrupt vectored locations must be filled with the corresponding instruction information. When repeated input or output operations are performed, only the byte count and buffer address must be respecified at the start of the operation.

```
*
* INTERRUPT VECTORED LOCATIONS (OUTPUT)
*
          ABS       INTAD               INTERRUPT VECTORED LOCATIONS
          AOB       GPDA                AUTOMATIC OUTPUT BYTE INSTRUCTION
          DATA      -OUTCNT             NEGATIVE OF THE OUTPUT BYTE COUNT
          BAC       OUTBUF-1            ADDRESS OF THE OUTPUT BUFFER
          DATA      0                   NOT USED
          JST       *$+1                CALL END-OF-BLOCK ROUTINE
          DATA      EOBLOC              ADDRESS OF END-OF-BLOCK ROUTINE


*
* INTERRUPT VECTORED LOCATIONS (INPUT)
*
          ABS       INTAD               INTERRUPT VECTORED LOCATIONS
          AIB       GPDA                AUTOMATIC INPUT BYTE INSTUCTION
          DATA      -INCNT              NEGATIVE OF THE INPUT BYTE COUNT
          BAC       INBUF-1             ADDRESS OF THE INPUT BUFFER
          DATA      0                   NOT USED
          JST       *$+1                CALL END-OF-BLOCK ROUTINE
          DATA      EOBLOC              ADDRESS OF END-OF-BLOCK ROUTINE
```

The following section of code is a subroutine used to transmit data. The routine first requests service by the other computer and then waits for a response before starting the data transmission. Once the data transmission is started, the subroutine returns to the caller. The actual data transmission continues until terminating conditions are detected, at which time the End-of-Block service is performed.

```
*
* SUBROUTINE TO OUTPUT DATA
*
          EQU       $
OUTDAT    ENT                           ENTRY POINT FOR SUBROUTINE
          LDA       =SETMOD+RQST        ISSUE REQUEST SERVICE TO THE OTHER COMPUTER
          OTA       GPDA+1              SEND COMMAND WORD TO I/O DISTRIBUTOR
NOTYET    INA       GPDA+1              INPUT STATUS FROM INTELLIGENT CABLE
          LRA       6                   ISOLATE SPARE (ACKNOWLEDGE) FLAG
          JOS       NOTYET              LOOP UNTIL ACKNOWLEDGE RECEIVED
          LDA       =SETMOD+BEGIN+OUT+WORD
                                        ASSEMBLE COMMAND WORD TO START DATA
                                        TRANSMISSION
          OTA       GPDA+1              SEND COMMAND WORD TO I/O DISTRIBUTOR
          EIN                           ENABLE INTERRUPTS
          RTN       OUTDAT              RETURN TO CALLER
*
* AUTOMATIC I/O BYTE COUNT AND BUFFER ADDRESS
*
OUTCNT    EQU       (specify)           LENGTH OF DATA TRANSMISSION
OUTBUF    EQU       (specify)           ADDRESS OF OUTPUT BUFFER
```

The following section of code is a subroutine used to accept data. The routine should be entered periodically to determine if the other computer is requesting to transmit data. If it is not, the subroutine is aborted and a return is made to the caller. If service is requested, the subroutine must assemble the interrupt vectored locations for input (code not shown). The subroutine then issues the command word to accept data through the 32-Bit G-P Intelligent Cable. This command word includes the response to the other computers service request. The subroutine then returns to the caller and data is accepted until terminating conditions are detected, at which time the End-of-Block service is performed.

```
*
* SUBROUTINE TO INPUT DATA
*
          EQU       $
INDAT     ENT                           ENTRY POINT FOR SUBROUTINE
          INA       GPDA+1              INPUT STATUS FROM INTELLIGENT CABLE
          LRA       6                   ISOLATE SPARE (ACKNOWLEDGE) FLAG
          JOR       ACKN                JUMP IF DATA INPUT REQUIRED
          RTN       INDAT               RETURN TO CALLER
ACKN       •                                    •
           •                            ASSEMBLE INTERRUPT VECTORED LOCATIONS
           •                            FOR INPUT
           •                                    •
          LDA       =SET MOD+BEGIN+IN+WORD+RQST
                                        ASSEMBLE COMMAND WORD TO ACCEPT DATA
          OTA       GPDA+1              SEND COMMAND WORD TO I/O DISTRIBUTOR
          RTN       INDAT               RETURN TO CALLER
*
* AUTOMATIC I/O BYTE COUNT AND BUFFER ADDRESS
*
INCNT     EQU       (specify)           LENGTH OF DATA INPUT
INBUF     EQU       (specify)           ADDRESS OF INPUT BUFFER
```

After all Automatic I/O data transfers are completed, the 32-Bit G-P Intelligent Cable vectors the last I/O interrupt to the End-of-Block service location. The subroutine called by this interrupt can be used to initiate processing of data transferred, perform error analysis, etc. The demonstration code simply halts.

```
*
* END-OF-BLOCK SERVICE
*
EOBLOC    ENT                           END OF BLOCK ENTRY POINT
          HLT
```

3.3.12.6.2 Card Punch

The 32-Bit G-P Intelligent Cable is especially useful for interfacing to peripheral devices, such as a card punch, where more than eight bits must be presented simultaneously to assure proper device operation. The interface shown in Figure 3-100 is representative of a card punch interface and with modification, this basic design could be used to connect an LSI Family computer to any one of several different card punches.

The minimum control line connections for a card punch interface are Function Strobe to initiate a punch cycle, Function Strobe Acknowledge to initiate reloading the buffer and Power to indicate the card punch is turned on. Successful operation is possible with this minimum control connection. Data/Command can be redefined as the Pick Card signal if a separate signal is required to initiate the pick operation. End can be redefined as the Eject Card signal to clear the card from the punch station after all the data is punched in the card. Stop can be redefined as Hopper Empty status to abort the operation if a supply of cards for punching is not available. Device Busy can be redefined as Card Not Registered status to assure that a card is registered in the punch station before a punch cycle is started. Spare can be redefined as Stacker Full status to indicate that a punched card can not be successfully stored in the card stacker.

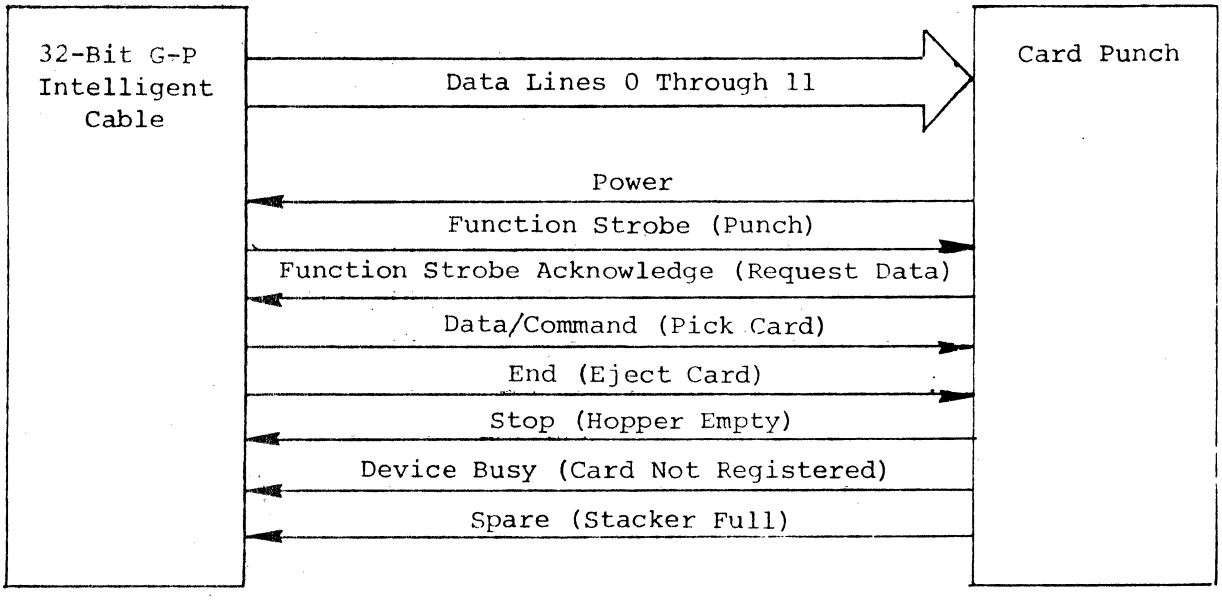



Figure 3-100. Hypothetical Card Punch Interface Overview

Programming Example

The Assembler Language statements shown in the following example demonstrate a basic program for using the 32-Bit G-P Intelligent Cable with a card punch. This example shows the various routines necessary to issue punch a card.

The first section of code is the parameter equates that define a series of constants used to assemble the various routines. The first five equates establish the device address and interrupt vector locations. The remaining equates select Intelligent Cable operations.

```
*
* PARAMETER EQUATES FOR 32-BIT G-P INTELLIGENT CABLE
*
INTBAS    EQU       :C0            INTERRUPT VECTOR BASE (STRAPPED ON I/O DISTRIBUTOR)
CHAN      EQU       (specify)      INTELLIGENT CABLE IS PLUGGED INTO THIS CHANNEL
IOD       EQU       :F             ADDRESS BASE (STRAPPED ON I/O DISTRIBUTOR)
INTAD     EQU       CHAN%3+INTBAS  DATA INTERRUPT VECTOR ADDRESS FOR INTELLIGENT CABLE
GPDA      EQU       IOD%3+CHAN%1   INTELLIGENT CABLE DEVICE ADDRESS
*
SETMOD    EQU       :400           SET MODE COMMAND WORD BASE
WORD      EQU       :1             SELECT 16-BIT WORD SIZE
PICK      EQU       :4             PICK CARD SIGNAL
BEGIN     EQU       :200           BRANCH COMMAND WORD BASE
OUT       EQU       :30            OUTPUT
```

The following section of code, interrupt vectored locations (output), defines and fills the memory locations associated with the I/O Distributor channel to which the 32-Bit G-P Intelligent Cable is connected. There are six memory locations involved with each of these sections of code; however, only the byte count and buffer address must be respecified at the start of each punch operation.

```
*
* INTERRUPT VECTORED LOCATIONS (OUTPUT)
*
          ABS       INTAD              INTERRUPT VECTORED LOCATIONS
          AOB       GPDA               AUTOMATIC OUTPUT BYTE INSTRUCTION
          DATA      -OUTCNT            NEGATIVE OF THE OUTPUT BYTE COUNT
          BAC       OUTBUF-1           ADDRESS OF THE OUTPUT BUFFER
          DATA      0                  NOT USED
          JST       *$+1               CALL END-OF-BLOCK ROUTINE
          DATA      EOBLOC             ADDRESS OF END-OF-BLOCK ROUTINE
```

The following section of code is a subroutine used to output data to the card punch. The routine first requests status and verifies that the card punch is supplied with card and ready to punch a card. The routine then initiates the card pick operation. Once the card is registered, the punching operation is started. The subroutine then returns to the caller while the punch operation continues until terminating conditions are detected and the End-of-Block service is performed.

```
*
* SUBROUTINE TO PUNCH ONE CARD
*
          EQU       $
PCARD     ENT                           ENTRY POINT FOR SUBROUTINE
          INA       GPDA+1              INPUT STATUS FROM INTELLIGENT CABLE
          LRA       4                   ISOLATE STOP (HOPPER EMPTY) FLAG
          JOR       ABRT                JUMP IF HOPPER EMPTY
          LRA       2                   ISOLATE SPARE (STACKER FULL) FLAG
          JOS       NPICK               JUMP IF STACKER NOT FULL
ABRT      HLT                           WAIT UNTIL ERROR IS CLEARED
NPICK     LDA       SETMOD+PICK         ASSEMBLE COMMAND WORD TO PICK CARD
          OTA       GPDA+1              SEND COMMAND WORD TO I/O DISTRIBUTOR
NOTYET    INA       GPDA+1              INPUT STATUS FROM INTELLIGENT CABLE
          LRA       3                   ISOLATE DEVICE BUSY (CARD NOT REGISTERED) FLAG
          JOR       NOTYET              LOOP UNTIL CARD IS REGISTERED
          LDA       SETMOD+BEGIN+OUT+WORD
                                        ASSEMBLE COMMAND WORD TO PUNCH CARD
          OTA       GPDA+1              SEND COMMAND WORD TO I/O DISTRIBUTOR
          EIN                           ENABLE INTERRUPTS
          RTN       PCARD               RETURN TO CALLER
*
* AUTOMATIC I/O BYTE COUNT AND BUFFER ADDRESS
*
OUTCNT    EQU       (specify)           TWO TIMES NUMBER OF CARD COLUMNS
OUTBUF    EQU       (specify)           ADDRESS OF OUTPUT BUFFER
```

After all Automatic I/O data transfers are completed, the 32-Bit G-P Intelligent Cable vectors the last I/O interrupt to the End-of-Block service location. The subroutine called by this interrupt can be used to initiate processing of data transferred, perform error analysis, etc. The demonstration code simply halts.

```
*
* END-OF-BLOCK SERVICE
*
EOBLOC    ENT                         END OF BLOCK ENTRY POINT
          HLT
```

### 3.3.12.6.3 Display Panel

The 32-Bit G-P Intelligent Cable can be used to interface to specially designed devices. The interface and device block diagram shown in Figure 3-101 represents one such special design peripheral device. The interface and block diagram shown here is for a 128 character display panel. The display itself is based on a 4-row by 32-column matrix. Additional display panel logic consists mainly of the following:

- A 32-bit data tranceiver section controlled by the Input/Output control line.
- A four character row driver-register. The register is loaded by the Step Pulse when data is output.
- A 32 step column locater shift register. The register count is advanced by the Step Pulse when data is output and loaded by the Step Pulse when command is output.

Both the row-driver register and the column locater shift register are initialized by reset. Following reset, the next data output is displayed in the leftmost column of the display board. Assuming a 30 Hz refresh rate is used to maintain a stable display, column updates must be output at 1 ms intervals.

Programming Example

The Assembler Language statements shown in the following example demonstrate a basic program for using the display panel application. This example shows the various routines necessary to operate the display panel.

The first section of code is the parameter equates that define a series of constants used to assemble the various routines. The first four establish the device address. The remaining equates select Intelligent Cable operations.

```
*
* PARAMETER EQUATES FOR 32-BIT G-P INTELLIGENT CABLE
*
INTBAS    EQU       :C0             INTERRUPT VECTOR BASE (STRAPPED ON I/O DISTRIBUTOR)
CHAN      EQU       (specify)       INTELLIGENT CABLE IS PLUGGED INTO THIS CHANNEL
IOD       EQU       :F              ADDRESS BASE (STRAPPED ON I/O DISTRIBUTOR
GPDA      EQU       IOD%3+CHAN%1    INTELLIGENT CABLE DEVICE ADDRESS
*
SETMOD    EQU       :400            SET MODE COMMAND WORD BASE
BYTE2     EQU       :1              SELECT 16-BIT WORD SIZE
BYTE3     EQU       :2              SELECT 24-BIT WORD SIZE
BYTE4     EQU       :3              SELECT 32-BIT WORD SIZE
WORD      EQU       BYTE 4          SELECTS THE WORD SIZE USED
BEGIN     EQU       :200            BRANCH COMMAND WORD BASE
INIT      EQU       :100            RESET COMMAND WORD
DATA      EQU       :4              SELECT DATA OUTPUT
STEP      EQU       :10             PULSE STEP FOR DATA OUTPUT
```

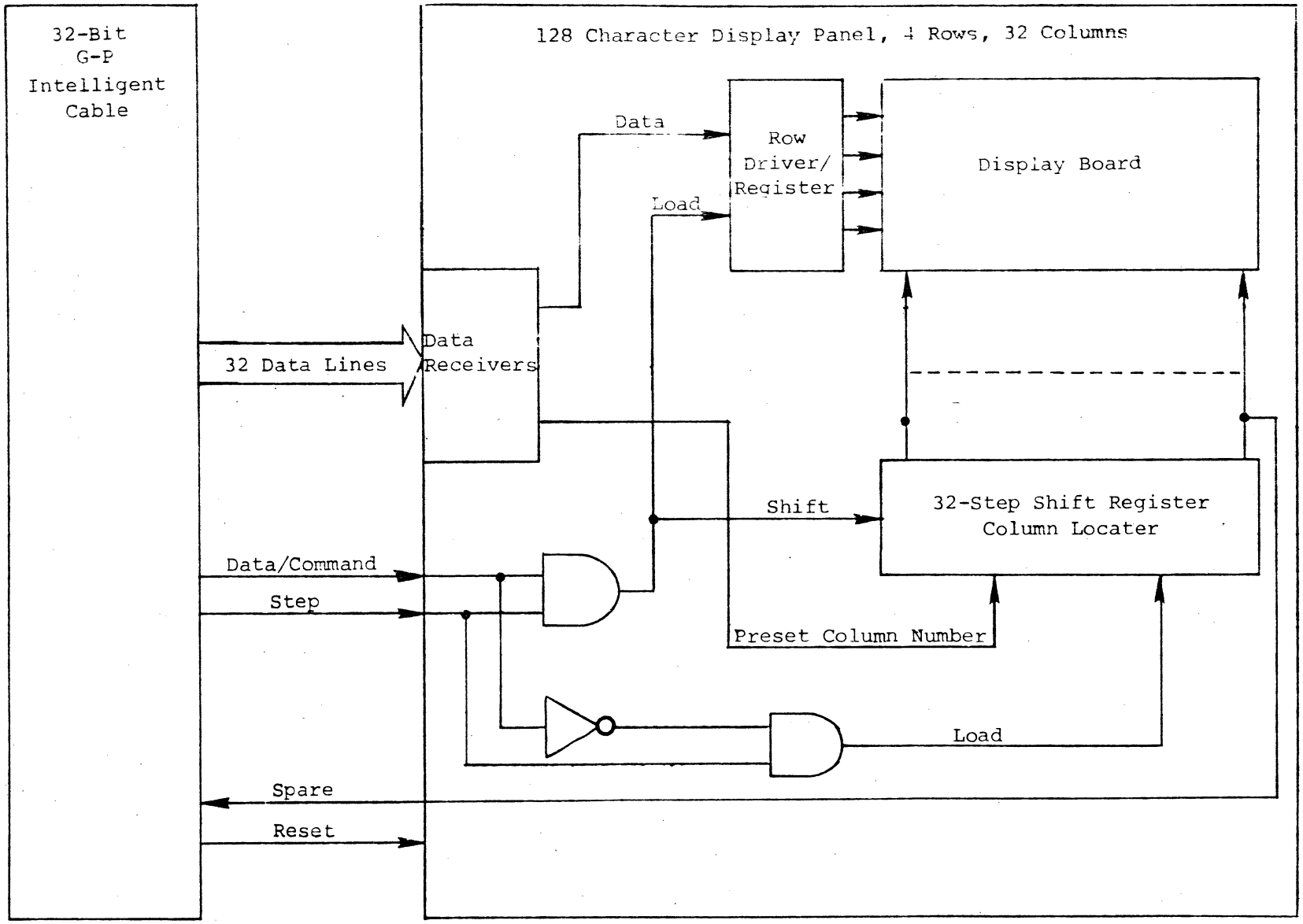



Figure 3-101. Hypothetical Display Panel Interface Overview

The following section of code is a routine used to display data on the display panel. The routine first initializes the 32-Bit G-P Intelligent Cable which also resets the display panel. The routine then outputs a word to the column locater to establish a starting point for the display of data. In the code shown, the column locater is preset to column 32 so the first data displayed is in column 1. Once the column locater is preset, the routine proceeds to output data for display. After each word is output, the code stores the incremented data buffer address (X register) and provision is noted for a software timer (code not shown) to develop the necessary one millisecond column update interval timing. After the timer interval, the code retrieves the contents of X register and then checks for the last column of the display being reached and if it is not, the next columns data is output. When the last column is reached, the code checks the sense switch for a signal to halt before returning for the next refresh cycle.

```
*
* ROUTINE TO DISPLAY DATA
* THE COLUMN LOCATER IS FIRST PRESET TO KNOWN POSITION
*
          REL       :200
DISPLAY   LDA       =INIT           COMMAND WORD TO INITIALIZE THE INTELLIGENT CABLE
          OTA       GPDA+1          SEND COMMAND WORD TO I/O DISTRIBUTOR
          LDA       =SETMOD+BYTE4   ASSEMBLE COMMAND WORD TO ESTABLISH PRESET WORD SIZE
          OTA       GPDA+1          SEND COMMAND WORD TO I/O DISTRIBUTOR
          LDX       =PRESET         LOAD PRESET BUFFER ADDRESS IN X REGISTER
          LLX       1               CONVERT TO BYTE ADDRESS
          JMP       PSTAT           GO TO INITIAL INTELLIGENT CABLE STATUS CHECK
          SBM                       SET BYTE MODE
NTFULA    LDAB      @0              LOAD PRESET BYTE
          SWM                       SET WORD MODE
          IXR                       INCREMENT INDEX
          OTA       GPDA            STORE BYTE IN INTELLIGENT CABLE BUFFER
PSTAT     INA       GPDA+1          REQUEST STATUS FROM INTELLIGENT CABLE
          LRA       5               ISOLATE BUFFER FULL FLAG
          JOS       NTFULA          RETURN FOR ADDITIONAL BYTES
          SBM                       SET BYTE MODE
          LDAB      @0              LOAD LAST PRESET BYTE
          SWM                       SET WORD MODE
          OTA       GPDA            STORE BYTE IN INTELLIGENT CABLE BUFFER
          LDA       =BEGIN+STEP     ASSEMBLE COMMAND WORD TO TRANSFER DATA
          OTA       GPDA+1          SEND COMMAND WORD TO I/O DISTRIBUTOR
*
* ROUTINE NOW PREPARES TO DISPLAY DATA
*
          LDA       =BEGIN+SETMOD+DATA+WORD
                                    ASSEMBLE COMMAND WORD TO ESTABLISH DATA WORD SIZE
          OTA       GPDA+1          SEND COMMAND WORD TO I/O DISTRIBUTOR
RFRESH    LDX       =DATBUF         LOAD DISPLAY DATA BUFFER ADDRESS IN X REGISTER
          LLX       1               CONVERT TO BYTE ADDRESS
          JMP       DSTAT           GO TO INITIAL INTELLIGENT CABLE STATUS CHECK
          SBM                       SET BYTE MODE
NTFULB    LDAB      @0              LOAD DISPLAY DATA BYTE
```

```
            SWM                         SET WORD MODE
            IXR                         INCREMENT INDEX
            OTA         GPDA            STORE BYTE IN INTELLIGENT CABLE BUFFER
DSTAT       INA         GPDA+1          REQUEST STATUS FROM INTELLIGENT CABLE
            LRA         5               ISOLATE BUFFER FULL FLAG
            JOS         NTFULB          RETURN FOR ADDITIONAL BYTES
            SBM                         SET BYTE MODE
            LDAB        @0              LOAD LAST DISPLAY DATA BYTE
            SWM                         SET WORD MODE
            IXR                         INCREMENT INDEX
            OTA         GPDA            STORE BYTE IN INTELLIGENT CABLE BUFFER
            LDA         =BEGIN+STEP     ASSEMBLE COMMAND WORD TO TRANSFER DATA
            OTA         GPDA+1          SEND COMMAND WORD TO I/O DISTRIBUTOR
            STX         XSAVE           STORE CONTENTS OF X REGISTER
             •                                      •
             •                          SOFTWARE TIMER TO PROVIDE ONE MILLISECOND
             •                          COLUMN UPDATE INTERVAL
             •                                      •
            LDX         XSAVE           RETRIEVE CONTENTS OF X REGISTER
            INA         GPDA+1          REQUEST STATUS FROM INTELLIGENT CABLE
            LRA         6               ISOLATE SPARE (COLUMN 32) FLAG
            JOS         DSTAT           RETURN FOR NEXT COLUMN UPDATE
            JSR         RFRESH          RETURN FOR NEXT DISPLAY PANEL REFRESH
            HLT
*
* PRESET AND DISPLAY DATA BUFFERS
*
PRESET      DATA        :0,:1
DATBUF      TEXT        (Specify)
```

# SECTION 4

# INSTALLATION

This section describes the complete installation of the Distributed I/O System for both standard and DMA I/O Distributors. All strapping requirements for the I/O Distributors are included. Strapping requirements for the Intelligent Cables are provided along with the individual Intelligent Cable descriptions in Section 3. Refer to the manuals provided by the peripheral manufacturer for any special installation requirements.

The installation sequence, as presented here, is the recommended order for performing the various steps. This sequence should be followed as closely as possible.

## 4.1 PRIORITY AND CHANNEL SELECTION

Each Intelligent Cable, when attached to an I/O Distributor, assumes a priority level. The location of this priority level, in relation to other peripherals, is determined by a combination of the following:

The type of I/O Distributor, standard I/O Distributor or DMA I/O Distributor, in use.

The priority level of the I/O Distributor channel to which the Intelligent Cable is attached, and the number and priority levels of the I/O Distributor's channels actually in use.

The priority level of the I/O Distributor in relation to other interface controllers attached to the computer.

When configuring a system for installation of an I/O Distributor(s), determine the desired priority level for each system peripheral device. Then plan the I/O Distributor(s) location within the computer and the channel assignments for the Intelligent Cables attached to each I/O Distributor. When determining priority levels, give special consideration to any special Intelligent Cable bandwidth requirements in relation to the bandwidth capabilities of the computer system used. The bandwidth specifications are listed in Section 1 of this manual.

### 4.1.1 I/O Distributor Channel Priority

The interrupt priority level of each Intelligent Cable attached to an I/O Distributor is determined by the channel number to which it is connected. The lowest numbered channel on an I/O Distributor has the highest priority. With the 8-channel standard I/O Distributor, the highest priority channel is Channel 0. With the 4-channel standard I/O Distributor, the highest priority channel is Channel 4. The DMA I/O Distributor is limited to four channels, either Channels 0 through 3 or Channels 4 through 7. Depending on the strapping, the priority channel is either 0 or 4.

The I/O Distributor does not scan the channels to service interrupt requests. When the I/O Distributor completes the service of an interrupt request, the highest priority interrupt request then present is serviced. All channel data service requests are assigned a higher priority by the I/O Distributor than any channel End-of-Block service request.

Computer Automation software uses the standard I/O Distributor channel assignments shown in Table 4-1. The device address shown assumes an eight-channel standard I/O Distributor with standard factory strapping for the I/O Distributor number. This list is included for information only and the use of these assignments is not required.

Table 4-1. Channel Assignments

| I/O Distributor Channel Number | Device Address | Intelligent Cable |
|---|---|---|
| 0 | :F0 | Card Reader |
| 1 | :F2 | Magnetic Tape |
| 2 | :F4 | Modem |
| 3 | :F6 | IEEE |
| 4 | :F8 | TTY or CRT |
| 5 | :FA | HSPT Reader |
| 6 | :FC | HSPT Punch |
| 7 | :FE | Line Printer |

### 4.1.2 Maxi-Bus Interrupt Priority

The interrupt priority level of each device attached to the computer Maxi-Bus is determined by its relative position in the interrupt priority chain. The device closest to the processor (in the interrupt priority chain) has the highest priority. Each additional device added to the interrupt priority chain assumes a lower priority determined by the total number of devices in the chain between it and the processor. With the Distributed I/O System, all four or eight channels of the I/O Distributor placed closest to the processor have a higher effective priority than any channel of and other I/O Distributor placed further from the processor. The priority level of an I/O Distributor is independent of the I/O Distributor address.

In the half-card chassis, the interrupt priority chain starts at the processor slot and each slot position further away from the processor assumes a lower priority.

In a full-card chassis, each slot is divided into a 100 side and a 200 side. The interrupt priority chain in these chassis starts at the half-card processor position, the 100 side of the top slot. (Facing the rear of the chassis, the 100 side is the right hand side.) The interrupt priority chain then goes to the 200 side of the top slot, down to the 200 side of the second slot, across to the 100 side of the second slot, down to the 100 side of the third slot etc. As before, each slot position further away from the processor assumes a lower priority.

### 4.1.3 Maxi-Bus DMA Priority

The DMA priority level of each DMA device attached to the computer Maxi-Bus is determined by its relative position in the DMA priority chain. As with the interrupt priority chain, the DMA device closest to the processor (in the DMA priority chain) has the highest priority. The DMA priority level of a DMA I/O Distributor is independent of the DMA I/O Distributor address.

NOTE

> Some DMA controllers have the ability to transfer data in bursts of two or more characters. After each character of the burst is transferred to memory, a higher priority DMA controller can interrupt the burst. Lower priority DMA controllers must wait until the burst in completed.

In the half-card chassis, the DMA priority chain starts at the processor slot and each slot position further away from the processor is a lower priority.

In a full-card chassis, the DMA priority chain starts at the half-card processor slot, the 100 side of the top slot. The DMA priority chain then goes to the 200 side of the top slot, down to the 200 side of the second slot, down to the 200 side of the third slot, etc. As before, each slot position further away from the processor is a lower priority.

The DMA I/O Distributor assumes a priority level in both the DMA priority chain and the interrupt priority chain. Since the interrupt priority level only applies to End-of-Block service, the placement of the DMA I/O Distributor in the interrupt priority chain has a minimum affect on the operation of the DMA I/O Distributor. However, the operation of standard I/O Distributors attached to the same Max-Bus may be affected. Placing a standard I/O Distributor further from the processor in the interrupt priority chain than a DMA I/O Distributor makes all priority levels assigned to the standard I/O Distributor a lower priority level than the DMA I/O Distributor's assigned End-of-Block interupt priority levels.

## 4.2 STANDARD I/O DISTRIBUTOR STRAPPING

The strapping on the standard I/O Distributor is used to assign the I/O Distributor number, select the interrupt vector block of addresses, and when the serial feature is used, select the baud clock rate for each channel. Figure 4-1 shows the strapping locations on the standard I/O Distributor.

### 4.2.1 Standard I/O Distributor Number Assignments

The decode logic of the standard I/O Distributor must be set to respond to the I/O Distributor address provided in Bits 4 through 7 of the instruction word. Refer to Figure 4-2. The ones in Bits 6 and 7 are part of the hardware and not selectable. The jumpering of Bits 4 and 5 complete the assignment of an address to the standard I/O Distributor.

The required jumpers are installed on the header provided at J2 using number 24 to 28 gauge solid wire and soldered connections. All the jumpers on J2 go straight across the header.

### 4.2.2 Interrupt Address Block Selection

The logic of the standard I/O Distributor must be set to provide the desired interrupt addresses. Refer to Figure 4-3. The interrupt address block consists of 64 consecutive memory positions. The channel service request provides Bits 3, 4, and 5 so that each channel is allocated eight consecutive memory locations. Bit 2 is controlled by the type of interrupt involved, data service or End-of-Block. The strapping options provided allows relocating the interrupt address block in increments of 64 addresses through a total of 512 memory addresses.

When a 4-channel standard I/O Distributor is used, only the 32 most significant consecutive memory addresses of an interrupt address block are used. The remaining 32 least significant memory addresses may be used for any other purpose including use by a DMA I/O Distributor as an interrupt address block. See Section 4.3 on DMA I/O Distributor Strapping for further information.

The required jumpers are installed on the header provided at J2 using number 24 to 28 gauge solid wire and soldered connections. All the jumpers on J2 go straight across the header.

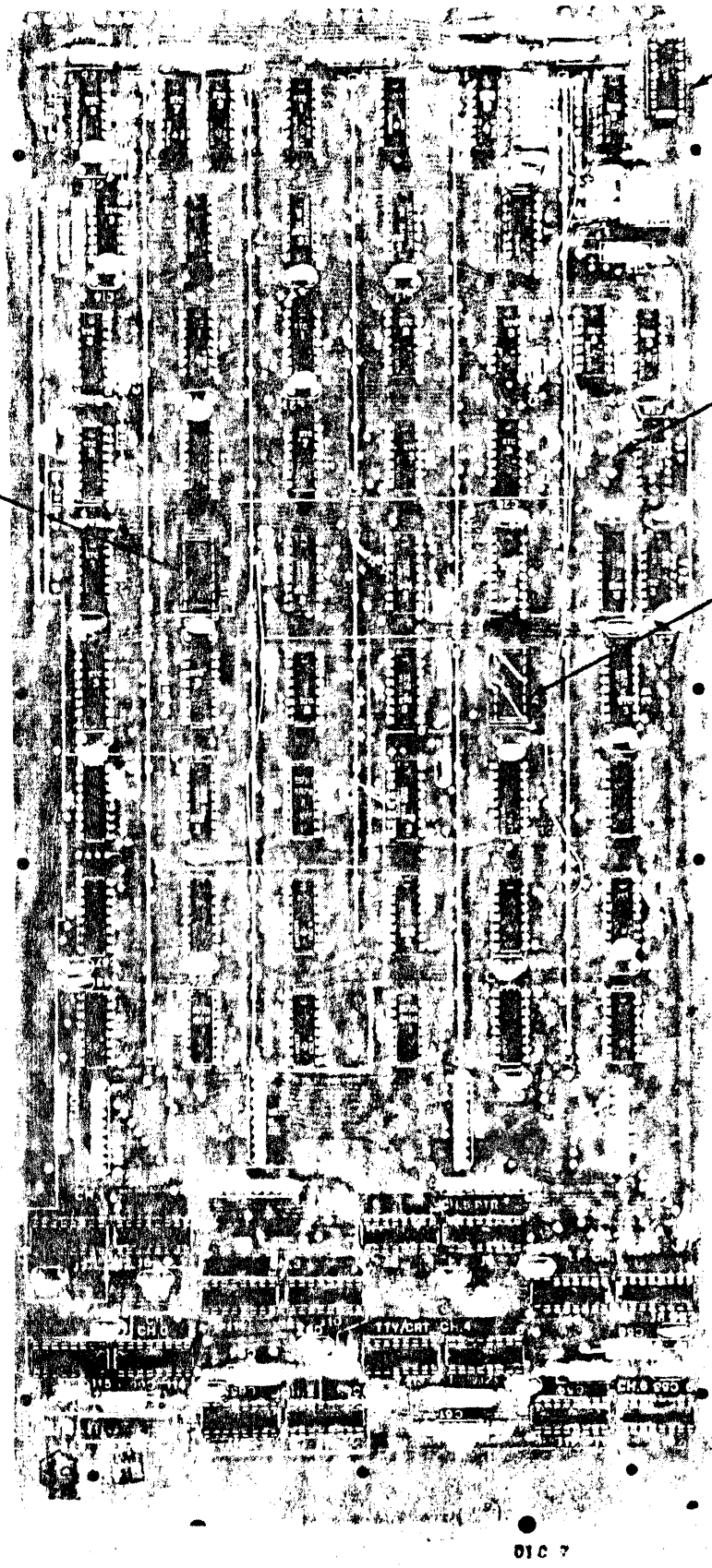
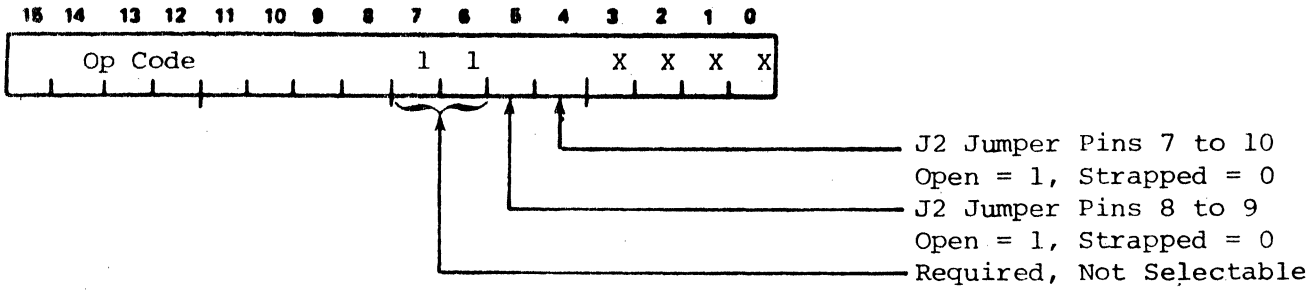



Figure 4-1. Standard I/O Distributor

I/O Instruction Word




I/O Distributor number assignment

| Jumper | | I/O Distributor | | Bit | Bit | |
|---|---|---|---|---|---|---|
| 8 to 9 | 7 to 10 | Number | Address | 5 | 4 | |
| In | In | 0 | :C | 0 | 0 | |
| In | Open | 1 | :D | 0 | 1 | |
| Open | In | 2 | :E | 1 | 0 | |
| Open | Open | 3 | :F | 1 | 1 | Standard |

J2
Configuration

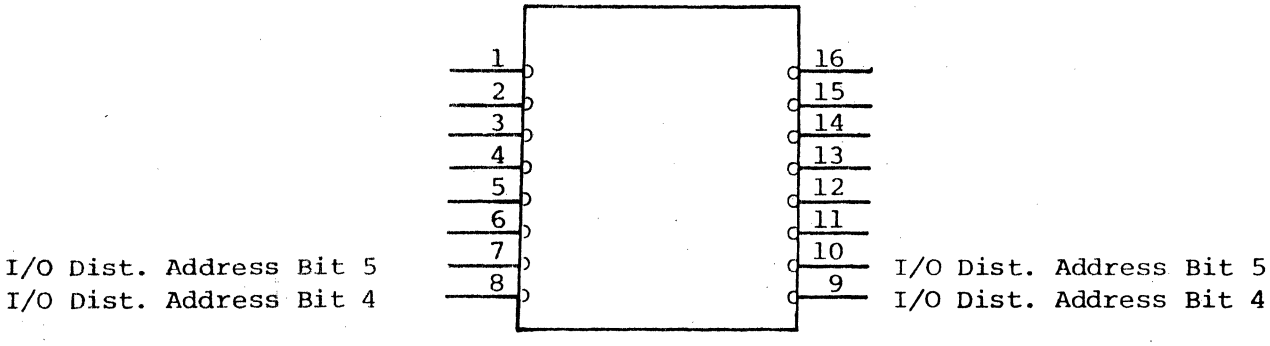
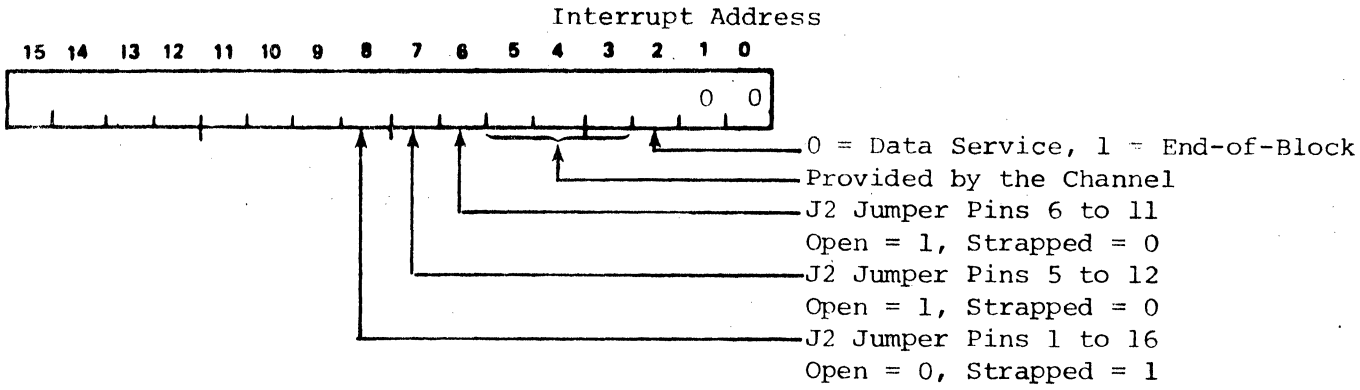



Figure 4-2. Standard I/O Distributor Address Strapping




Interrupt Address Block Selection

| Interrupt Address Block | Jumper | | | |
|---|---|---|---|---|
| | 1 to 16 | 5 to 12 | 6 to 11 | |
| :00 through :3F | Open | In | In | |
| :40 through :7F | Open | In | Open | |
| :80 through :BF | Open | Open | In | |
| :C0 through :FF | Open | Open | Open | Standard |
| :100 through :13F | In | In | In | |
| :140 through :17F | In | In | Open | |
| :180 through :1BF | In | Open | In | |
| :1C0 through :1FF | In | Open | Open | |

J2 Configuration

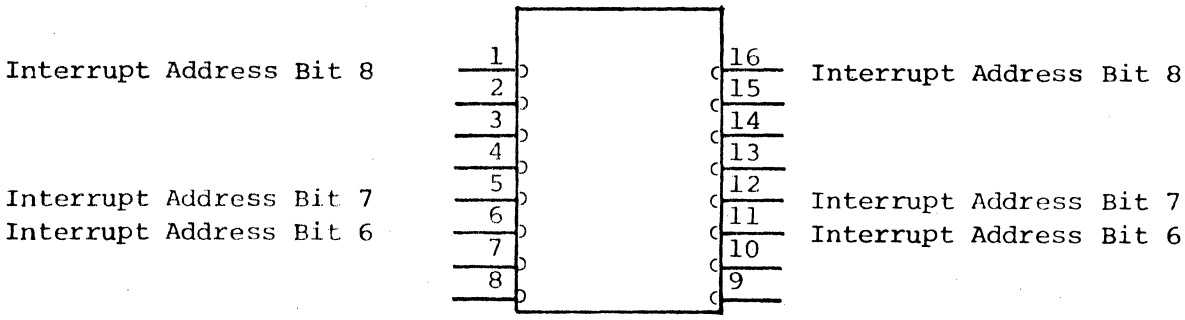



Figure 4-3. Standard I/O Distributor Interrupt Address

### 4.2.3 Baud Rate Selection

Baud rate selection is required when one or more serial data format Intelligent Cables are attached to a standard I/O Distributor. Ten separate baud rates are provided. Of these, seven are directly available for use by any of the I/O Distributor's channels. Refer to Figure 4-4. One of the three remaining rates, 75 baud, 600 baud, and 19,200 baud, can be selected for inclusion with the seven directly available rates. This makes up a set of eight baud rates. Each channel can then be independently strapped to any of the available baud rates. The J3 header is factor wired to select a baud rate of 110 for Channel 4.

The required jumpers are installed on the header provided at J3 using number 24 to 28 gauge insulated wire and soldered connections. If the 75 baud, 600 baud, or 19,200 baud rate is required, a jumper is installed between the "D" pad and the pad (A, B, or C) corresponding to the desired baud rate. The rate selected is available at pin 10 on J3.

J3 Configuration

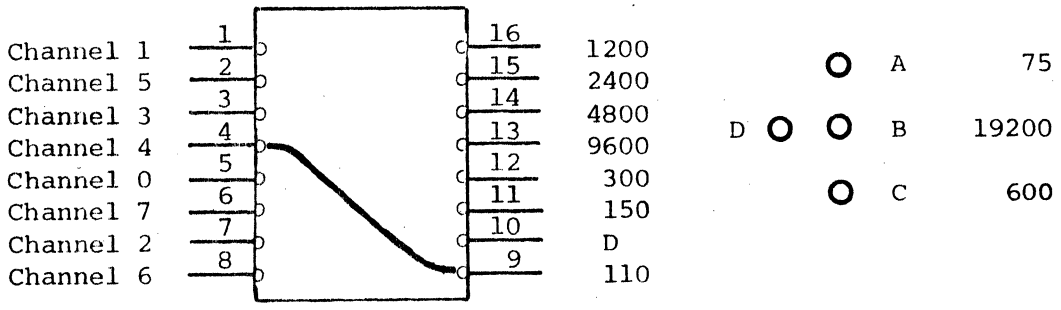



Figure 4-4. Standard I/O Distributor Baud Rate Strapping

## 4.3 DMA I/O DISTRIBUTOR STRAPPING

The strapping on the DMA I/O Distributor is used to assign the I/O Distributor number, select the interrupt vector Block of addresses, select the baud clock rate for each channel, and select the parity standardization bit value. Figure 4-5 shows the strapping locations on the DMA I/O Distributor.

### 4.3.1 DMA I/O Distributor Number Assignments

The decode logic of the DMA I/O Distributor must be set to respond to the I/O Distributor address provided in Bits 4 through 7 of the instruction word. Refer to Figure 4-6. The ones in Bits 6 and 7 are required and are not selectable. The jumpering of Bits 4 and 5 complete the assignment of an address to the I/O Distributor.

An additional address jumper assigns the four DMA I/O Distributor channels as 0 through 3 or 4 through 7. This feature allows a DMA I/O Distributor to share an interrupt address block with a 4-channel standard I/O Distributor or with another DMA I/O Distributor. When two I/O Distributors share an interrupt address block, both I/O Distributors should be assigned the same number and address. The two I/O Distributors can then be treated as a single, 8-channel I/O Distributor.

The required jumpers are installed on the header provided at J2 using number 24 to 28 gauge solid wire and soldered connections. All the jumpers on J2 go straight across the header.

### 4.3.2 Interrupt Address Block Selection

The logic of the DMA I/O Distributor must be set to provide the desired interrupt addresses for End-of-Block service. The addresses that correspond to the data service interrupt addresses in the standard I/O Distributor are used by the DMA I/O Distributor when the Auto I/O instruction is fetched or restored. Refer to Figure 4-7.

The interrupt address block consists of 64 consecutive memory positions. The channel End-of-Block service request provides Bits 3, 4, and 5 so that each channel is allocated eight consecitive memory locations. The DMA I/O Distributor only interrupts for End-of-Block service so Bit 2 is always a 1 for the interrupt. (Bit 2 is a 0 for the fetch and restore operation. The strapping options provided allows relocating the interrupt address block in increments of 64 addresses through a total of 512 memory addresses.

The DMA I/O Distributor is limited to four channels and only 32 consecutive memory addresses of the block of 64 are used. If the DMA I/O Distributor is strapped for channel 0 through 3, the 32 least significant memory addresses in the block are used. If the DMA I/O Distributor is strapped for channels 4 through 7, the 32 most significant memory addresses in the block are used. The remaining 32 memory addresses in the block may be used for any other purpose including use by another DMA I/O Distributor or a 4-channel standard I/O Distributor as an interrupt address block.

The required jumpers are installed on the header provided at J2 using number 24 to 28 gauge solid wire and soldered connections. All the jumpers on J2 go straight across the header.

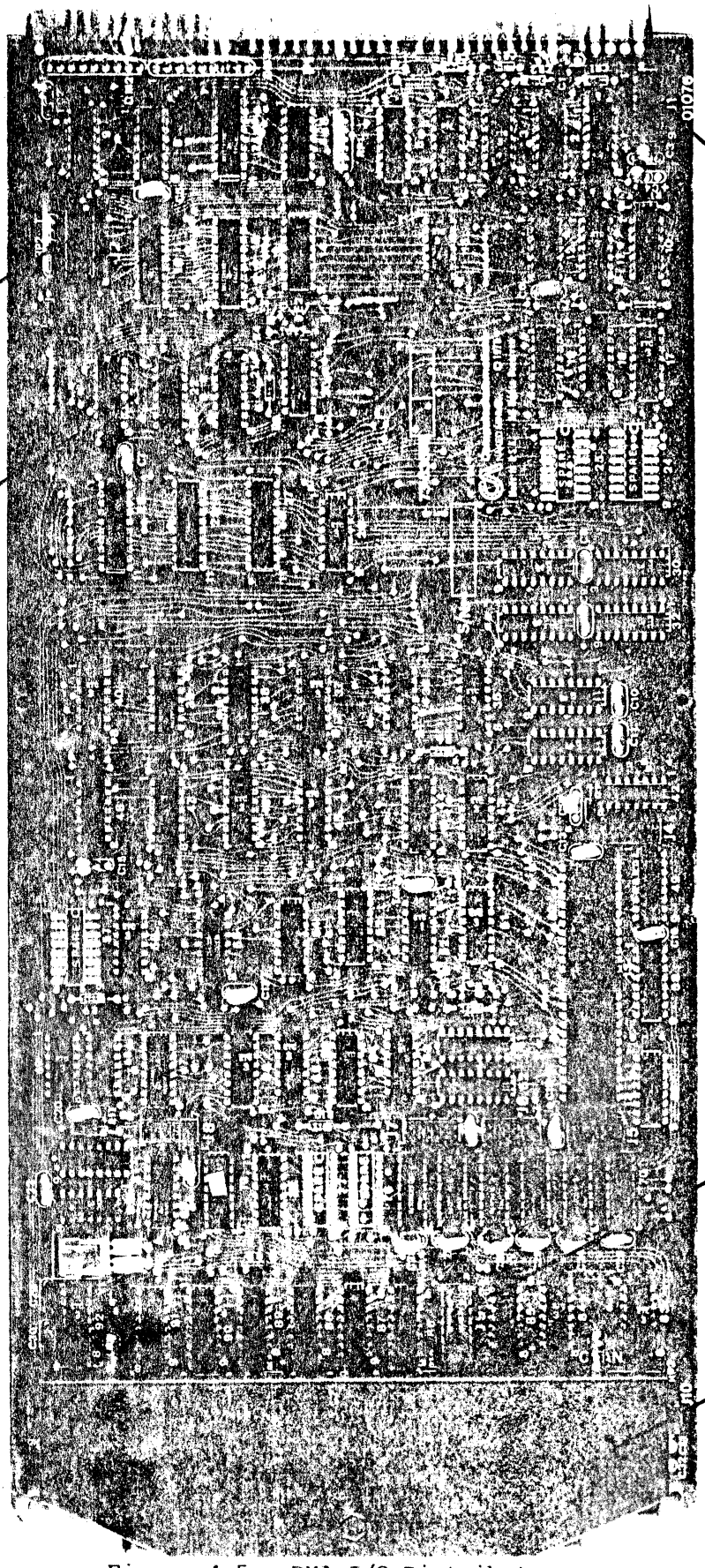
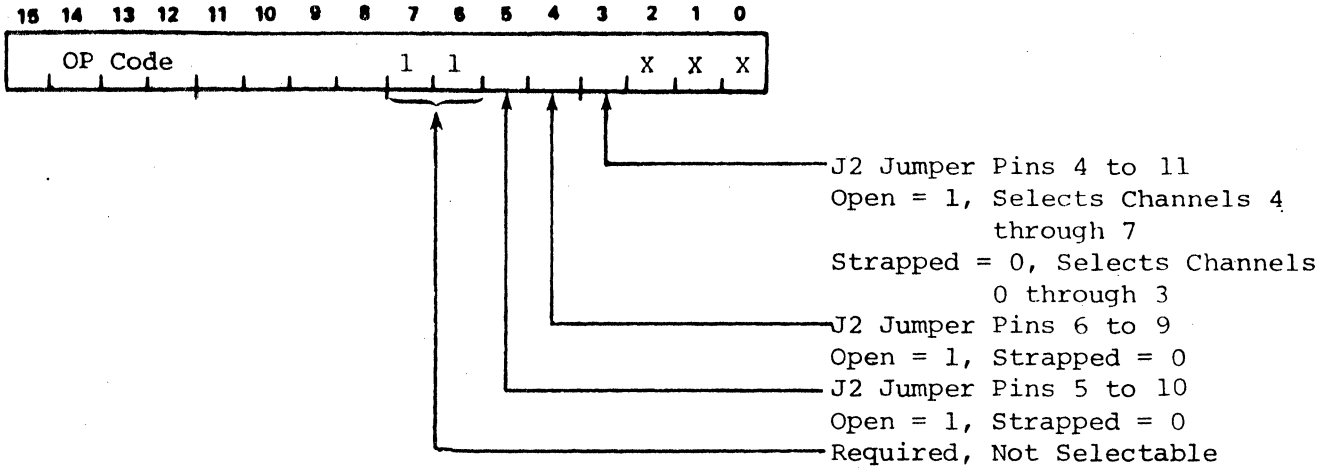



Figure 4-5. DMA I/O Distributor

I/O Instruction Word




I/O Distributor Number Assignment

| Jumper | | I/O Distributor | | Bit | Bit | |
|---|---|---|---|---|---|---|
| 5 to 10 | 6 to 9 | Number | Address | 5 | 4 | |
| In | In | 0 | :C | 0 | 0 | |
| In | Open | 1 | :D | 0 | 1 | |
| Open | In | 2 | :E | 1 | 0 | |
| Open | Open | 3 | :F | 1 | 1 | Standard |

J2 Configuration

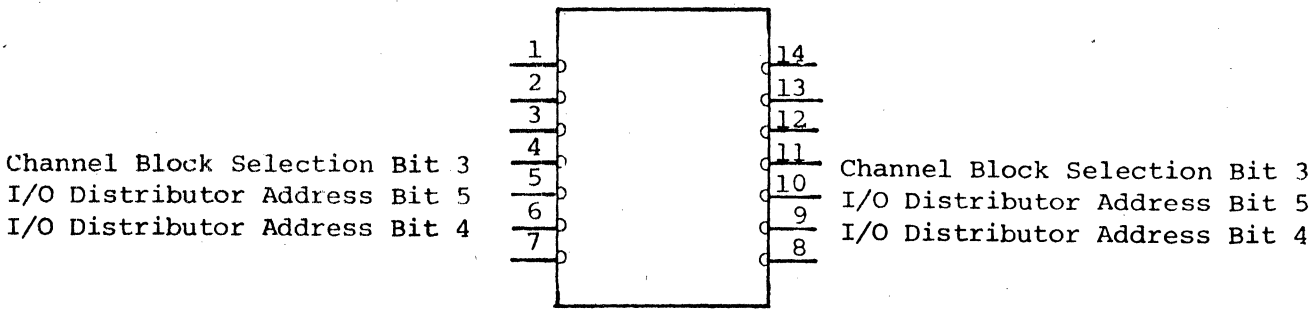
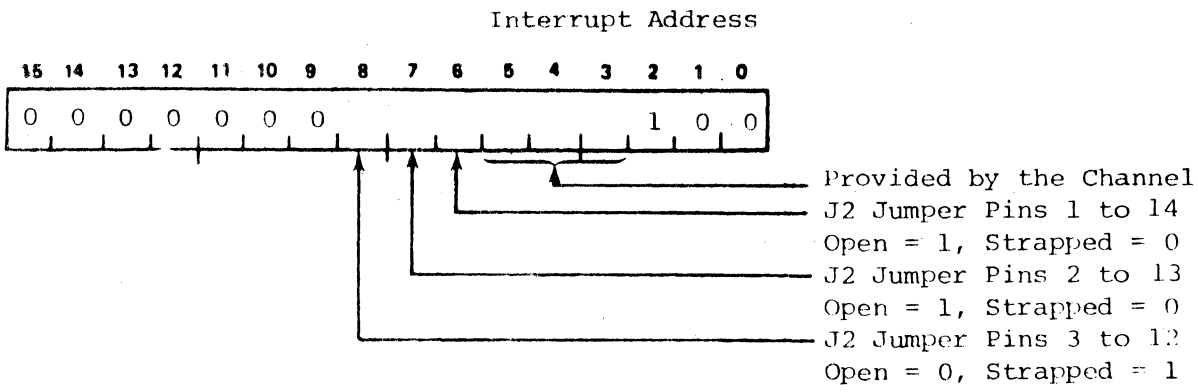



Figure 4-6. DMA I/O Distributor Address Strapping




Interrupt Address Block Selection

| Interrupt Address Block | Jumper | | | |
|---|---|---|---|---|
| | 3 to 12 | 2 to 13 | 1 to 14 | |
| :00 through :3F | Open | In | In | |
| :40 through :7F | Open | In | Open | |
| :80 through :BF | Open | Open | In | |
| :C0 through :FF | Open | Open | Open | Standard |
| :100 through :13F | In | In | In | |
| :140 through :17F | In | In | Open | |
| :180 through :1BF | In | Open | In | |
| :1C0 through :1FF | In | Open | Open | |

J2 Configuration

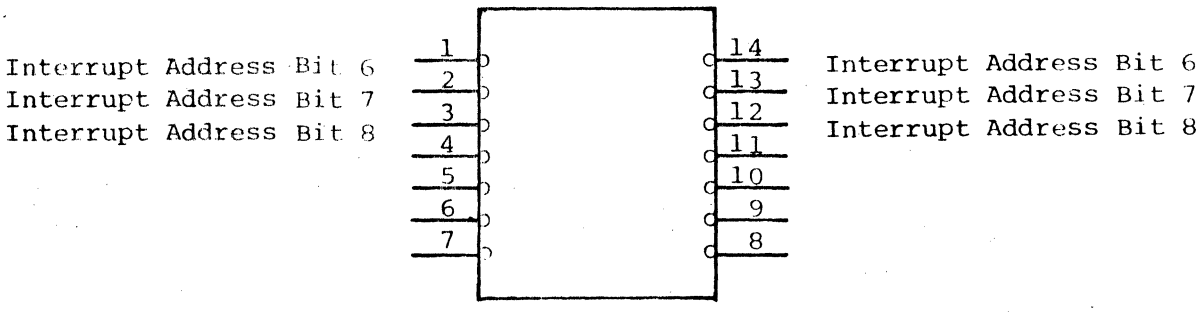



Figure 4-7. DMA I/O Distributor Interrupt Address

### 4.3.3 Baud Rate Selection

Baud rate selection is required when one or more serial data format Intelligent Cable/peripheral pairs are attached to a DMA I/O Distributor. The ten separate baud rates provided are all directly available for use by any of the DMA I/O Distributors channels. Refer to Figure 4-8.

The required jumpers are installed on the header provided at J9 using number 24 to 28 gauge insulated solid wire, and soldered connections.

J9 Configuration

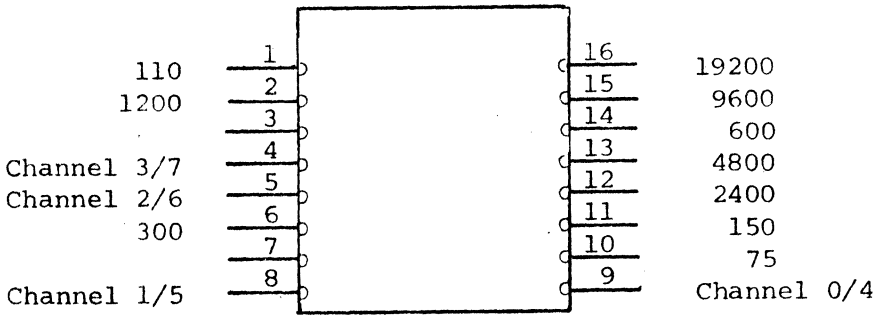



Figure 4-8. DMA I/O Distributor Baud Rate Strapping

### 4.3.4 Parity Standardization

Parity Standardization selection is available to allow use of either a "1" or a "0" as the substitution value of the parity. The DMA I/O Distributor is factory strapped for a "1" as the substitution value. To select the "0", remove the jumper between E1 and E2. Install a jumper between E2 and E3 using number 24 to 28 gauge solid wire and soldered connection.

## 4.4 I/O DISTRIBUTOR INSTALLATION

The installation of the I/O Distributor involves several operations. The type of I/O Distributor and the chassis used determines the specific operations required. This section describes the installation of both the DMA and standard I/O Distributor into the half-card chassis. The procedures for joining two half-cards together and the installation of both I/O Distributors into a full-card chassis is also described.

### 4.4.1 Standard I/O Distributor, Half-Card Chassis

The standard I/O Distributor installation kit for a half-card chassis includes an Intelligent Cable retainer/strain relief and the necessary mounting hardware. Refer to Figure 4-9. Prepare the standard I/O Distributor for installation as follows:

1. Install both circuit card ejectors on the I/O Distributor using the split pins provided. Position the ejectors so the short arms extend beyond the sides of the circuit card.

2. Install the stiffener so the raised portion is closest to the outer edge of the circuit card. Attach the stiffener with the hardware provided.

3. Plug the I/O Distributor end of the Intelligent Cables into the desired channel connectors.

4. Ensure that the ribbon cables on the outer row of Intelligent Cables are flat and smooth from the connector to the stiffener and that the ribbon cables on the inner row of Intelligent Cables are flat and smooth from the connector, across the outer row of connectors to the stiffener.

5. Place the cable retainer over all the Intelligent Cables so that all five holes in the cable retainer align with the mating holes in the mounting blocks or stiffener. Attach the cable retainer with the hardware provided. The standard I/O Distributor is now ready for installation in a half-card chassis.

### 4.4.2 DMA I/O Distributor, Half-Card Chassis

The DMA I/O Distributor is shipped with a hinged cable retainer assembled to the circuit card. Refer to Figure 4-10 and prepare the DMA I/O Distributor for installation as follows:

1. Remove the screw securing the outer edge of the cable retainer. Pivot the cable retainer on the hinges to expose the channel connectors.

2. Install two circuit card ejectors on the I/O Distributor (if required) using the split pins provided. Position the ejectors so the short arms extend beyond the sides of the circuit card.

3. Plug the I/O Distributor end of the Intelligent Cables into the desired channel connectors.

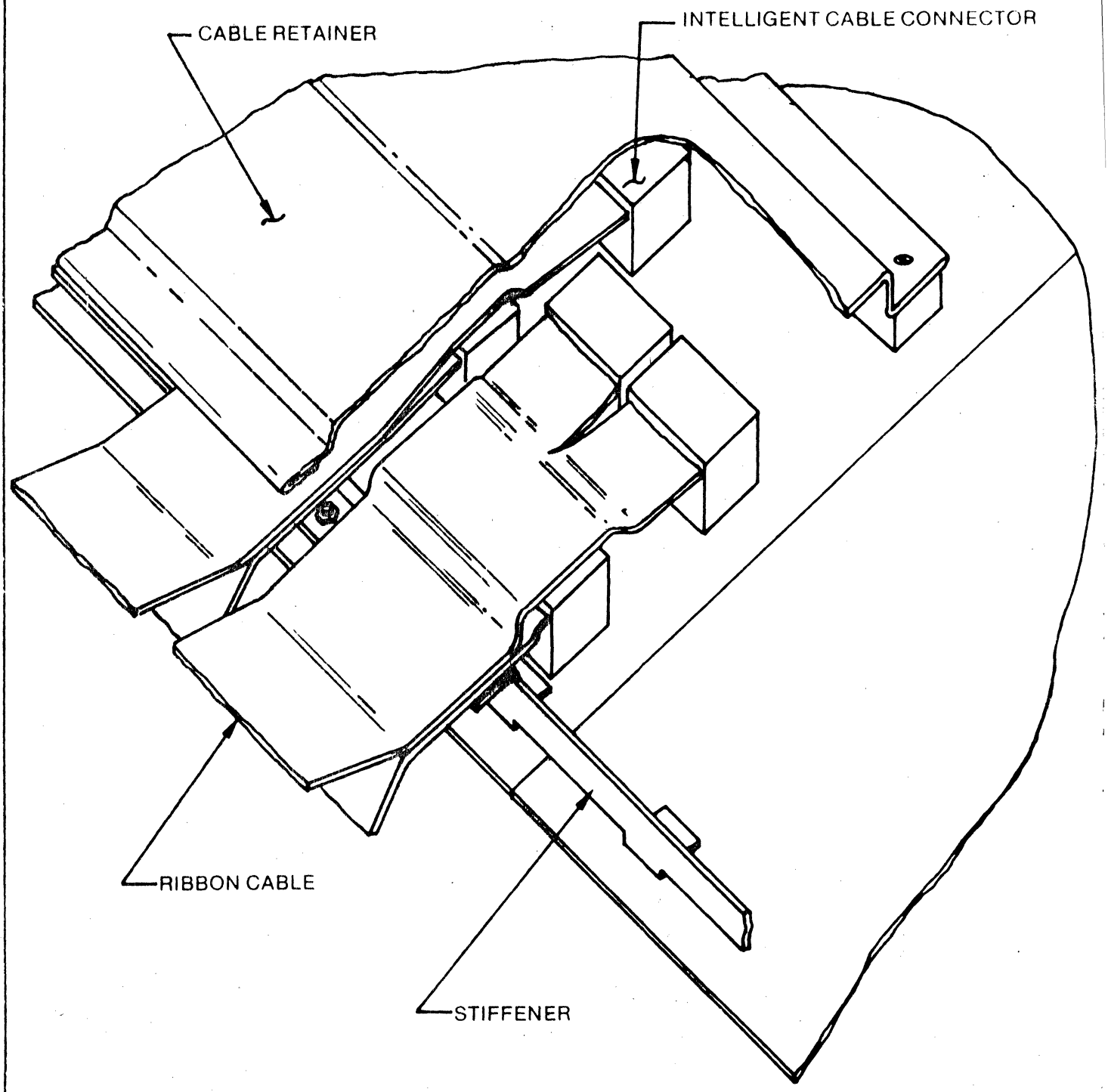



Figure 4-9. Standard I/O Distributor Cable Retainer (shown joined to another half-card)

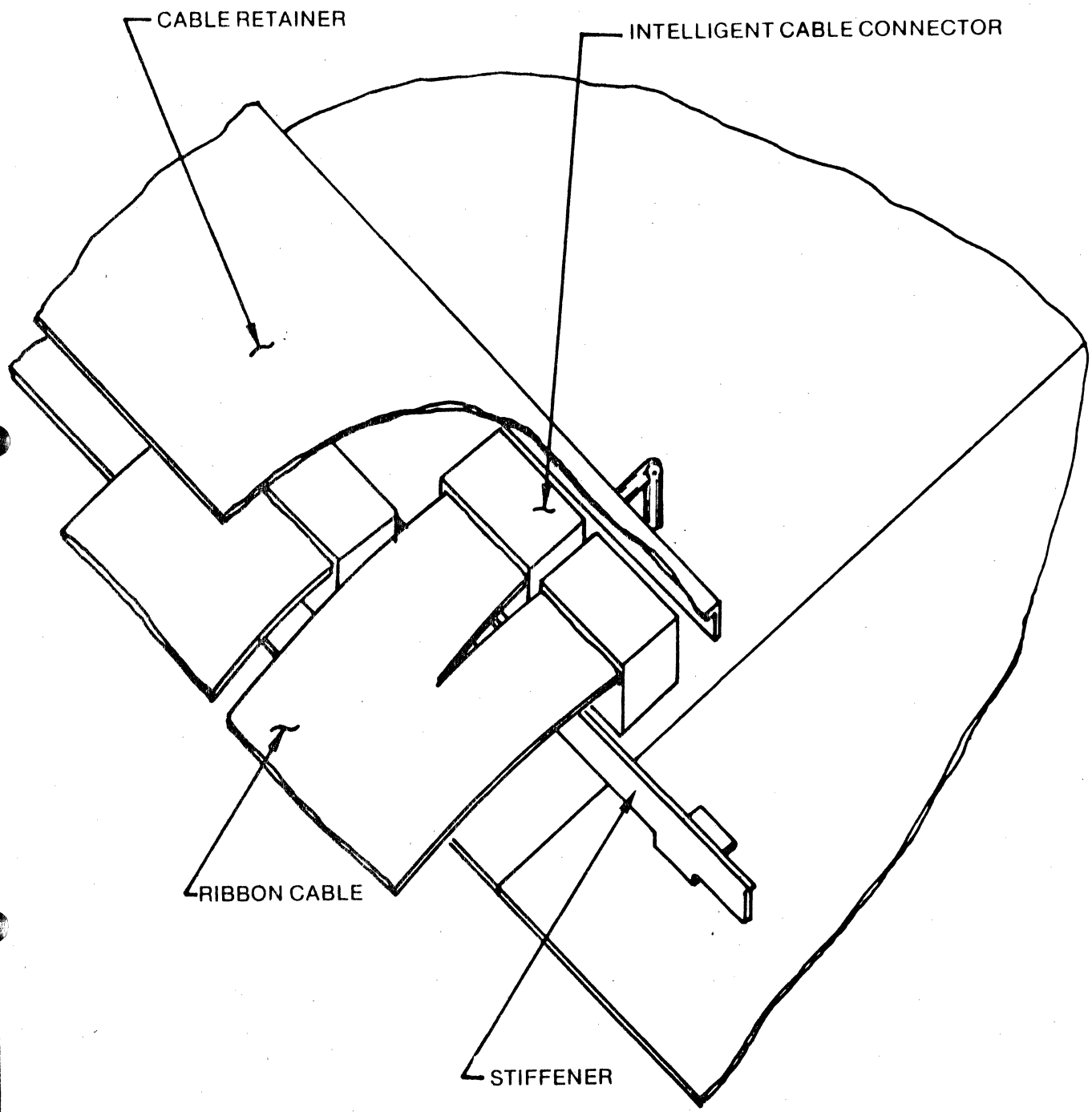



Figure 4-10. DMA I/O Distributor Cable Retainer
(shown joined to another half-card)

4. Pivot the cable retainer down over the Intelligent Cable connectors. Replace the screw removed in Step 1. The DMA I/O Distributor is now ready for installation in a half-card chassis.

### 4.4.3 I/O Distributor Location, Full-Card Chassis

When the I/O Distributor is installed in a full card chassis, the half-card width I/O Distributor must be joined to another half-card device or a filler half-card to produce the required full-card width.

The decision of chassis location and the mating device for the I/O Distributor is controlled primarily by the I/O Distributor's assigned priority level. The chassis location and choice of mating device must also conform to the requirements listed here and illustrated in Figure 4-11.

- Any I/O Distributor can be joined, as the 200 side, to a half-card processor. The assembled full-card must be installed in slot A of the computer (not expansion) chassis.

- The standard I/O Distributor can be joined, as the 100 side, to a half-card memory, DMA controller, or DMA I/O Distributor. The assembled full-card can be installed in any slot except A in the computer chassis or any slot in the expansion chassis.

- The DMA I/O Distributor can be joined, as the 200 side, to a standard I/O Distributor or filler half-card. The assembled full-card can be installed in any slot except slot A in the computer chassis or any slot in the expansion chassis.

- The standard I/O Distributor can be joined to either side of a filler half-card. The assembled full-card can be installed in any slot except slot A in the computer chassis or any slot in the expansion chassis.

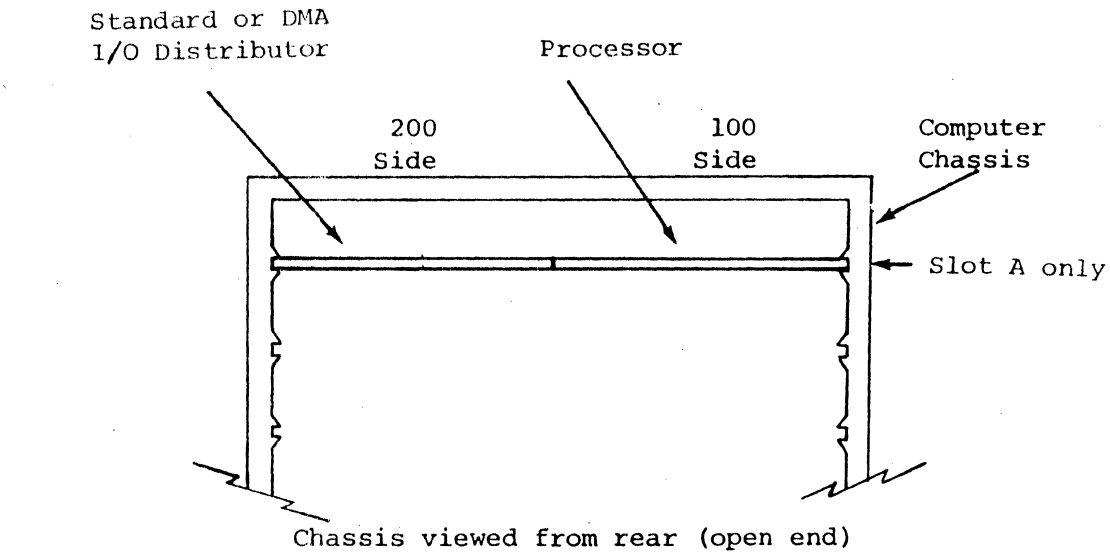
NOTE

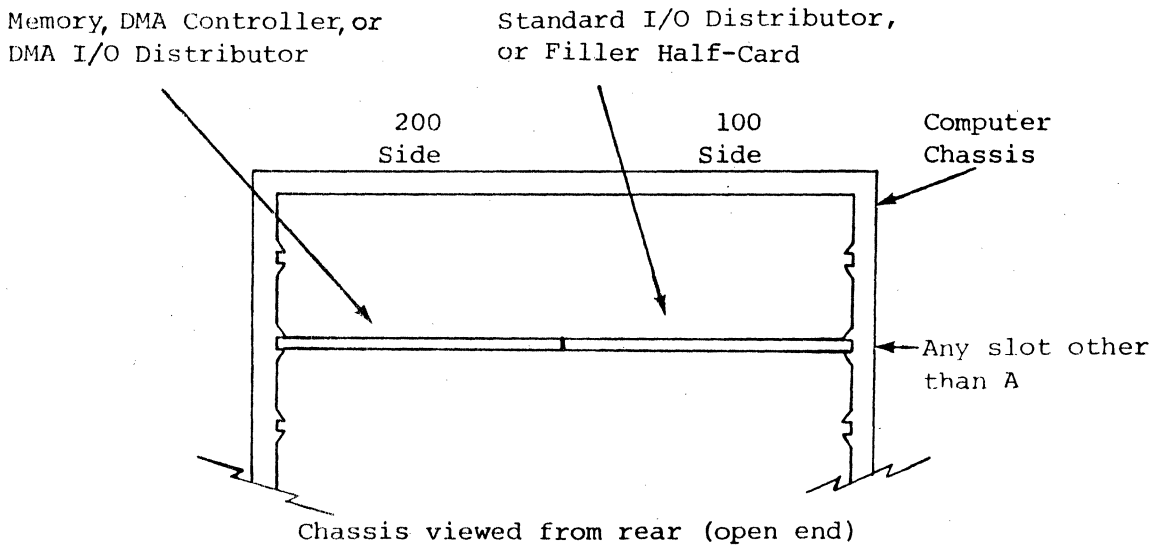
The standard I/O Distributor cannot be joined to another standard I/O Distributor. A full-card stiffener bar to join two standard I/O Distributors together is not available.

### 4.4.4 Standard I/O Distributor, Full-Card Chassis

The standard I/O Distributor installation kit for a full-card chassis is a universal kit. It allows assembly of the full-card with the standard I/O Distributor on either the 100 or 200 side of the chassis. This kit contains all necessary hardware for joining two half-cards together, including two combination stiffener bars. One end of each bar is combined with an Intelligent Cable retainer/strain relief. Prepare the standard I/O Distributor for installation as follows:

1. Determine to which side of the standard I/O Distributor the second half-card is to be attached. To this side of the standard I/O Distributor, attach the tapped plastic mounting blocks using the nylon screws. Do not fully tighten these screws. The mounting blocks must be on the component side of the circuit card.



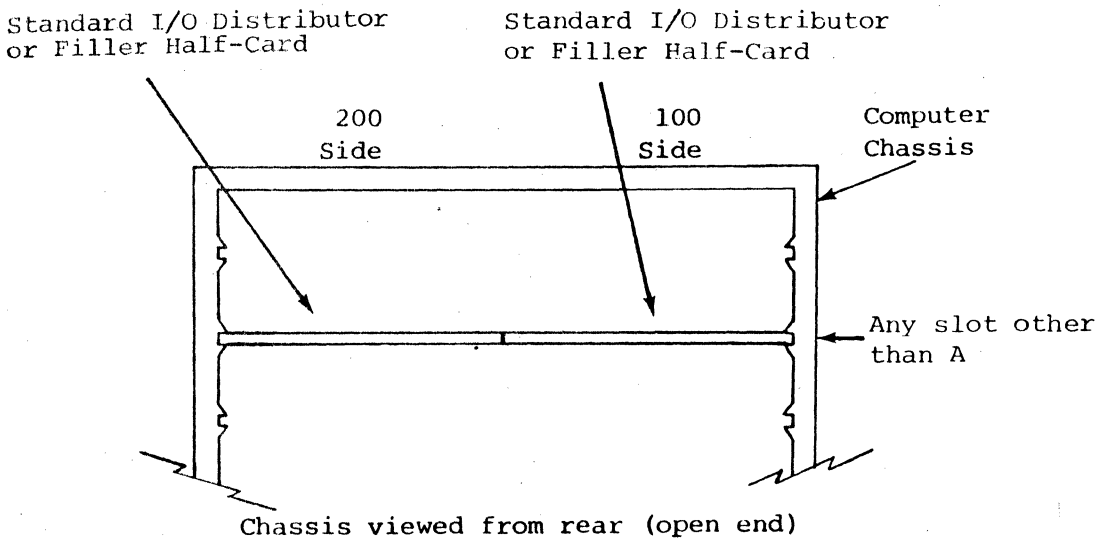




Figure 4-11. Chassis Layout

2. Select one of the stiffener bars provided. Attach this stiffener bar to the component side of the standard I/O Distributor at the channel connector end with the hardware provided. Do not fully tighten these screws. The portion of the stiffener bar attached to the standard I/O Distributor must be the end with the Intelligent Cable retainer/strain relief. The unattached end of the stiffener bar must be on the same side of the circuit card as the plastic mounting blocks.

3. Attach the second half-card to the mounting blocks (nylon screws) and the stiffener bar. Fully tighten all screws.

4. Install the two circuit card ejectors at the outside edges of the assembled full-card using the split pins provided. Position the ejectors so the short arms extend beyond the sides of the circuit card.

5. When the standard I/O Distributor is attached to a half-card processor, install the flat ribbon cable jumper (provided with the processor) between J1 on the standard I/O Distributor and the adjacent connector on the processor.

6. Plug the I/O Distributor end of the Intelligent Cables into the desired channel connectors.

7. Ensure that the ribbon cables on the outer row of Intelligent Cables are flat and smooth from the connector to the stiffener and the ribbon cables on the inner row of intelligent cables are flat and smooth from the connector, across the outer row of connectors, to the stiffener.

8. Place the cable retainer over all the Intelligent Cables so that all five holes in the cable retainer align with the mating holes in the mounting blocks or stiffener. Attach the cable retainer with the hardware provided. The standard I/O Distributor is not ready for installation in a full-card chassis.

### 4.4.5 DMA I/O Distributor, Full-Card Chassis

The DMA I/O Distributor installation kit is the standard kit used to join two half-cards together.

1. On the DMA I/O Distributor, remove the screw securing the outer edge of the cable retainer. Pivot the cable retainer on the hinges to expose the channel connectors.

2. To the right side of the DMA I/O Distributor, attach the tapped plastic mounting blocks using the nylon screws. Do not fully tighten these screws. The mounting blocks must be on the component side of the circuit card.

3. Attach the stiffener bar to the component side of the DMA I/O Distributor at the channel connector end with the hardware provided. Do not fully tighten these screws. The unattached end of the stiffener bar must be on the right side of the circuit card.

4. Attach the second half-card to the mounting blocks (nylon screws) and the stiffener bar. Fully tighten all screws.

5. Install the two circuit card ejectors at the outside edges of the assembled full-card using the split pins provided. Position the ejectors so the short arms extend beyond the sides of the circuit card.

6. When the DMA I/O Distributor is attached to a half-card processor, install the flat ribbon cable jumper (provided with the processor) between J1 on the DMA I/O Distributor and the adjacent connector on the processor.

7. Plug the I/O Distributor end of the Intelligent Cables into the desired channel connectors.

8. Make sure the Intelligent Cable's ribbon cable is smooth from the connector to over the stiffener bar.

9. Pivot the cable retainer down over the Intelligent Cable connectors. Replace the screw removed in Step 1. The DMA I/O Distributor is now ready for installation in a full-card chassis.

## 4.5 PICOPROCESSOR INSTALLATION

The Intelligent Cable is delivered with an adhesive backed Velcro fastener attached to the PICOPROCESSOR. Mount the PICOPROCESSOR on the peripheral device, cabinet, or equipment rack as follows:

1. Separate the outer strip of Velcro from the strip attached to the PICOPROCESSOR.

2. Ensure that the desired mounting surface is clean and dry and free of oil residue.

3. Peel the backing sheet off the adhesive on the Velcro strip removed in Step 1.

4. Press the adhesive side of the Velcro to the desired mounting surface.

5. Complete the installation by pressing the Velcro strip on the PICOPROCESSOR

Four mounting holes are provided in the PICOPROCESSOR case for hardware mounting if such a mounting is required. Figure 4-12 shows the mounting hole pattern.

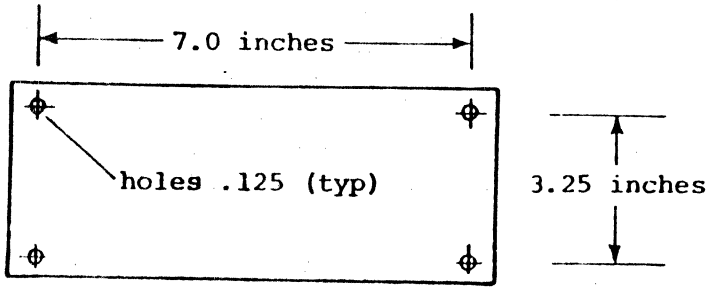



Figure 4-12. Chassis Layout

User's Manual
Distributed I/O System
91-53629-00
Supplement 1
September, 1975

# ADDENDUM

## Intelligent Cables

### 1.1 INTRODUCTION

This addendum describes Intelligent Cables not included in section 3 of the Distributed I/O System User's Manual. Listed below are the Intelligent Cables described and the corresponding peripheral devices.

| Intelligent Cable | Peripheral Device |
|---|---|
| 14631-53 | Trend Paper Tape Reader Model UDR 350/700 or equivalent |
| 14631-54 | Facit Paper Tape Punch Model 4070 or equivalent |

### 2.1 TREND PAPER TAPE READER

#### 2.1.1 Description

Intelligent Cable 14631-53 controls the transfer of eight-bit negative-true parallel data from a Trend Model UDR 350/700 Paper Tape Reader to the computer at the rate of 350 or 700 characters per second. The cable employs the same PICOPROCESSOR and software as Intelligent Cable 14631-03 (described in paragraph 3.3.3 of the Distributed I/O System User's Manual) and differs from that cable only at its device end. The PICOPROCESSOR accepts a start command from the IOD, checks device status, then issues a Data Service interrupt to start the data transfer. When the transfer is complete or an error detected, the PICOPROCESSOR generates an end-of-block interrupt to terminate the transfer.

#### 2.1.2 Physical Details

Cable Lengths:
IOD to PICOPROCESSOR, 4 feet
PICOPROCESSOR to Reader, 1-1/2 feet.

Standard Channel Number, 5 (Device Address field = :FA)

Standard Data Service Interrupt Address, :E8

Standard End-of-Block Interrupt Address, :EC

### 2.1.3 Paper Tape Reader Status Word

To control the transfer of data from the paper tape reader, the PICOPROCESSOR sequences through a series of operations based, in part, on the state of individual bits of the Paper Tape Reader Status Word. The Status Word, shown in figure 2-1, indicates certain operational conditions in the paper tape reader. When the PICO-PROCESSOR receives an input instruction requesting device status (function control bit set to "1"), it immediately transfers the entire Status Word to the CPU on bits 0 through 5 of the data bus. Input status can be requested at any time, but it is usually done after an end-of-block interrupt to determine the reason for termination. Individual bits of the Status Word are described in paragraph 2.1.5.2.

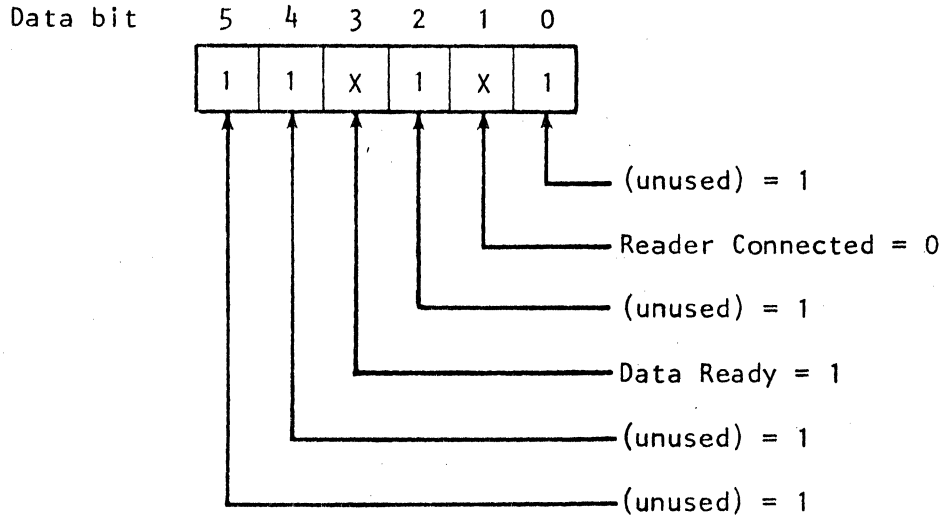



Figure 2-1. Trend Paper Tape Reader Status Word

### 2.1.4 Operating Sequence

The PICOPROCESSOR has 16 unique sequence addresses (:0 thru :F). When the PICO-PROCESSOR receives a Command Word with the Begin bit set to "1", it immediately begins operation at the sequence address specified by the Branch Address field of the Command Word. Standard software enters the sequence at one of the following sequence addresses.

| Sequence Address | Operation |
|---|---|
| :1 | Read Tape, Right to Left |

Details of the PICOPROCESSOR operating sequence are shown in the flow chart (figure 2-2). Interface lines are described in paragraph 2.1.5. Following is a sequence description with each segment of the operation identified by name and sequence address (:0 thru :F). The yes/no decisions refer to the true/false state of a particular line as defined in paragraph 2.1.5, regardless of the logic level.

IDLE (:0)

The PICOPROCESSOR is in the Idle state as a result of a Reset command or because of the completion of an end-of-block interrupt. A Begin command selects a Start sequence for a Drive Left Operation.

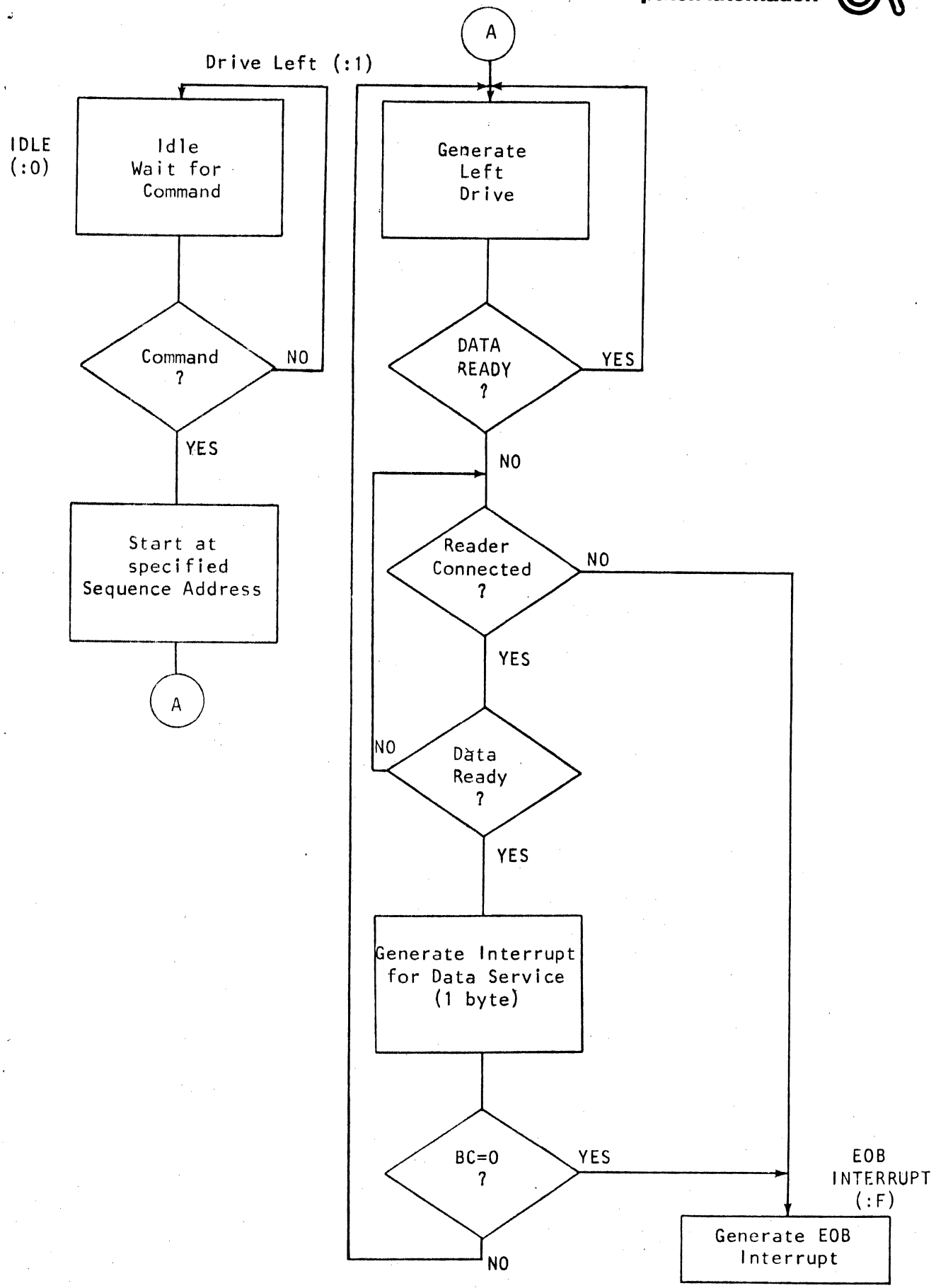



Figure 2-2. Firmware Sequence - Trend Reader Intelligent Cable

START LEFT (:1)

A Drive Left command is issued to start tape motion and Data Ready status is monitored. When the Data Ready line goes false, the reader head is moving and "between characters" on the tape. Reader Connected status is then checked. This line is true if the reader is connected. If false, the PICOPROCESSOR immediately generates an end-of-block interrupt to terminate the operation. Otherwise, the PICOPROCESSOR monitors the Data Ready status line again, this time looking for a true level on the line to indicate that the reader head is "on character". When a positive response is obtained, the PICOPROCESSOR generates a Data Service interrupt, causing the Automatic input instruction at the data interrupt location to be executed. Data from the reader is transferred into the CPU and the byte count and memory buffer byte address are incremented. A line from the CPU is activated if the byte count increments to zero to signal the PICOPROCESSOR (via the IOD) that all data has been transferred (end-of-block).

If the byte count does not increment to zero, the PICOPROCESSOR again generates Left Drive, checks the status of Data Ready, checks Reader Connected, and generates another Data Service Interrupt. This operation is repeated until all data has been transferred. The PICOPROCESSOR then enters the EOB Sequence.

EOB INTERRUPT (:F)

The PICOPROCESSOR generates an EOB interrupt to terminate operations when all data has been transferred or when a status error is detected at any point in the sequence. When the EOB interrupt is serviced, the CPU commands the PICOPROCESSOR (via the IOD) to return to the Idle state and wait for the next command.

### 2.1.5 Interface Description

Interface lines between the PICOPROCESSOR and the Trend Reader consist of eight data lines, one control line to the reader and two status lines from the reader to the PICOPROCESSOR (figure 2-3).

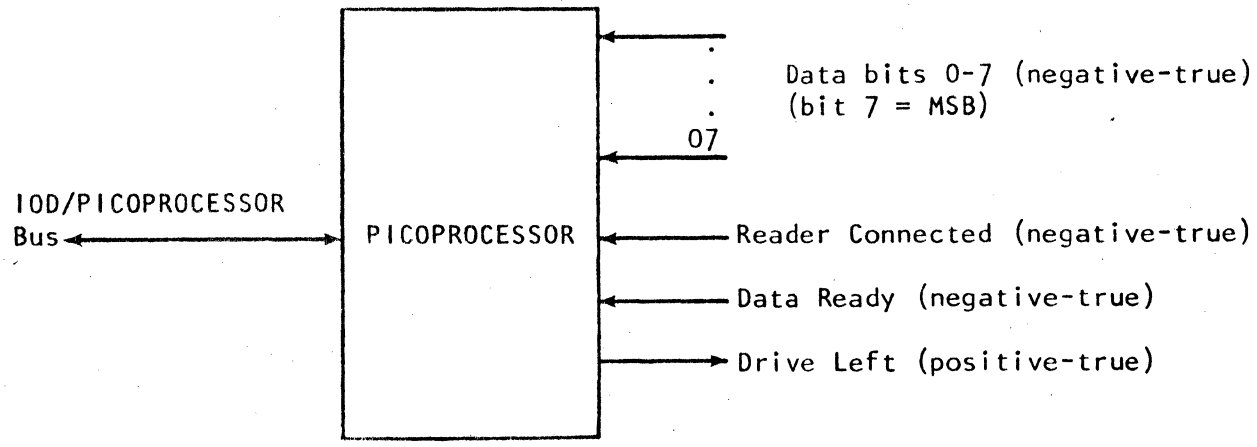



Figure 2-3. Trend Paper Tape Reader Interface

2.1.5.1 Control Lines (to Device)

1. Drive Left. This line (positive-true) is driven by the Intelligent Cable to start the tape moving in the reader in the left-hand direction.

2. Reset. This negative-true line is driven by the CPU RESET switch or under software control. Under software control, it is a 250-ns pulse. It is not used by the paper tape reader.

2.1.5.2 Status Lines (from Device)

1. Data Ready (negative-true). This line, when true, indicates that the data track outputs are in the "on character" position. When false, this line indicates the "between character" position where data outputs have no significance.

2. Reader Connected (negative-true). This line is connected to chassis ground and indicates that the reader is connected.

Other status bits are not used by the paper tape reader.

2.1.5.3 Data Input Lines

The eight data input lines (00 through 07) are negative-true. Bit 07 is the most-significant bit. The data is input from the reader in standard eight-bit ASCII characters.

Figure 2-4 shows the interface timing. For additional details, see the Trend Paper Tape Reader instruction manual.

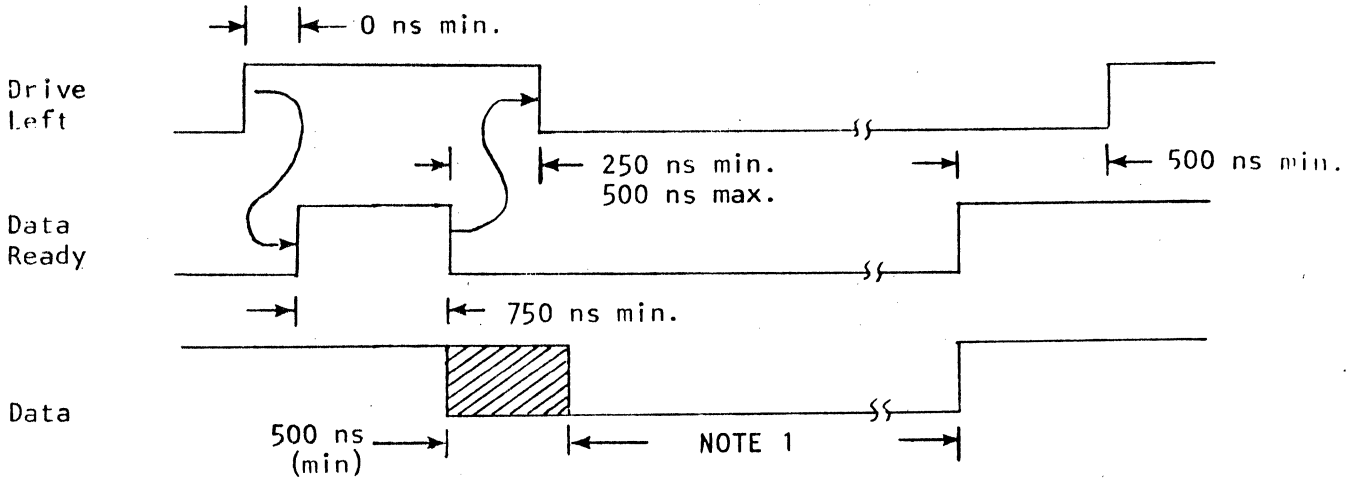
2.1.6 Strapping Requirements

(None Required.)

2.1.7 Device Cable Description

The Device cable is 18 inches long and terminated on the PICOPROCESSOR end with three 16-pin DIP plugs (P4, P5, and P6). The paper tape reader end of the cable is terminated with a paddle board that mates with connector SKT1 on the reader.

Figure 2-6 lists all interface lines in the device cable and identifies the connectors used. The location of mating connectors on the PICOPROCESSOR is shown in figure 2-5.




NOTE 1: Minimum time equal to the instruction time of the Auto I/O byte instruction plus 4 $\mu$s. For instruction time, see the appendix of the appropriate Computer Handbook.

Figure 2-4. Interface Timing - Trend Paper Tape Reader Intelligent Cable

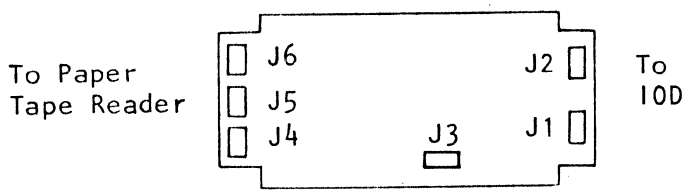



Figure 2-5. Connector Locations, Trend Reader PICOPROCESSOR

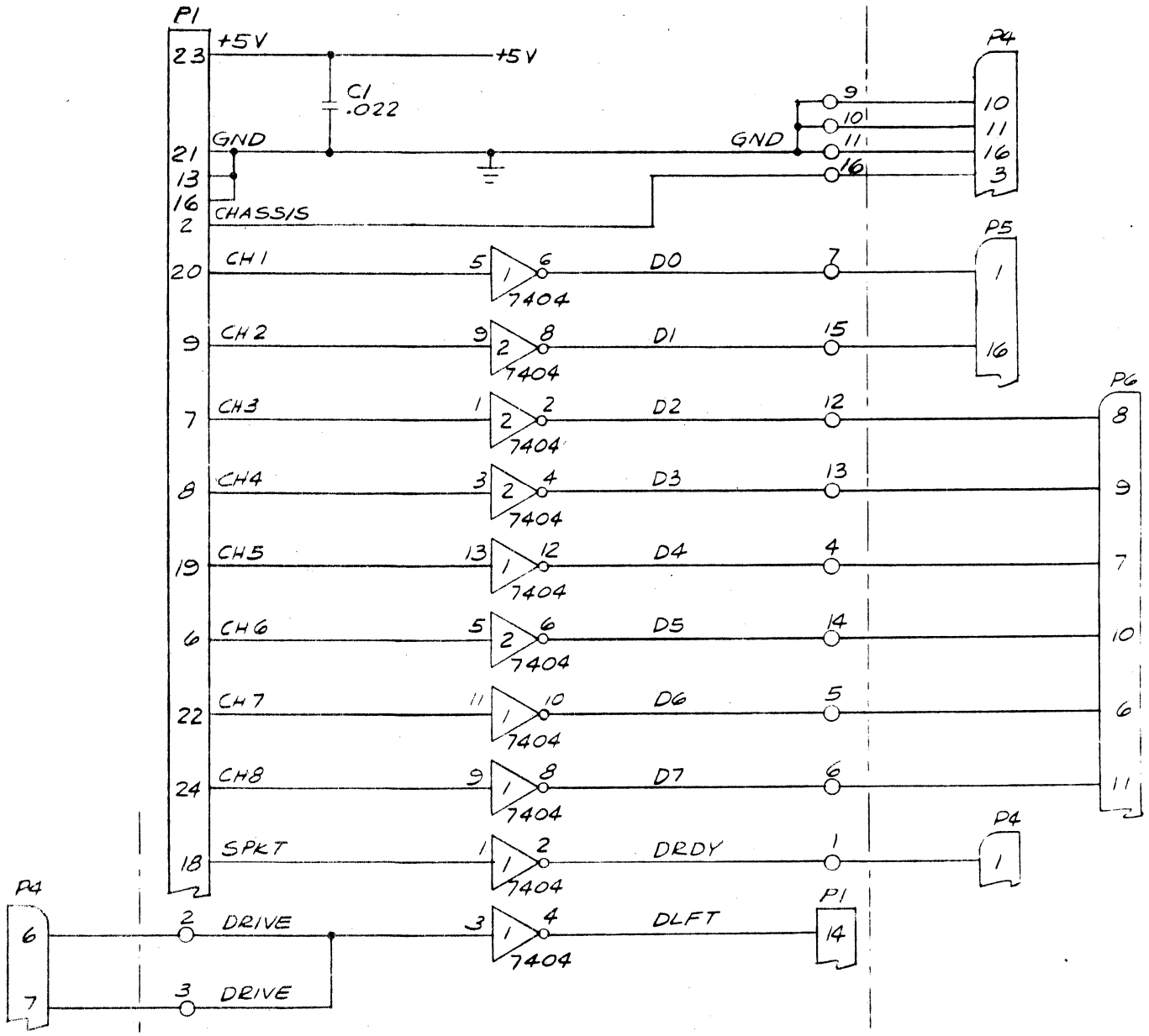
| PICOPROCESSOR Conn | PICOPROCESSOR Pin | Description | Paddle Board |
|---|---|---|---|
| P4 | 1 | Data Ready | E1 |
| ↑ | 2 | Status bit 2 (not used) | -- |
| ↕ | 3 | System Ready (chassis ground) | E16 |
| ↕ | 4 | Status line 0 (not used) | -- |
| ↕ | 5 | Reset (not used) | -- |
| ↕ | 6 | Drive | E2 |
| ↕ | 7 | Drive | E3 |
| ↕ | 8 | Control line 0 (not used) | -- |
| ↕ | 9 | Ground (not used) | -- |
| ↕ | 10 | Ground | E9 |
| ↕ | 11 | Ground | E10 |
| ↕ | 12 | Ground (not used) | -- |
| ↕ | 13 | Ground (not used) | -- |
| ↕ | 14 | Ground (not used) | -- |
| ↓ | 15 | Ground (not used) | -- |
| P4 | 16 | Ground | E11 |
| P5 | 1 | Data bit 00 | E7 |
| ↑ | 2 thru | } Not used | -- |
| P5 | 6 | | |

| PICOPROCESSOR Conn | PICOPROCESSOR Pin | Description | Paddle Board |
|---|---|---|---|
| P5 | 7 | Status bit 5 (not used) | -- |
| ↑ | 8 | Status bit 4 (not used) | -- |
| ↕ | 9 thru 15 | } Not used | -- |
| P5 | 16 | Data bit 01 | E15 |
| P6 | 1 | | |
| ↑ | thru 5 | Not used | -- |
| ↕ | 6 | Data bit 06 | E5 |
| ↕ | 7 | Data bit 04 | E4 |
| ↕ | 8 | Data bit 02 | E12 |
| ↕ | 9 | Data bit 03 | E13 |
| ↕ | 10 | Data bit 05 | E14 |
| ↕ | 11 | Data bit 07 | E6 |
| ↕ | 12 thru 15 | } Not used | -- |
| P6 | 16 | Ground (not used) | -- |

(Note the paddle board at the reader end of the cable; the paddle board contains signal inversion logic which permits the use of the standard paper tape reader PICOPROCESSOR and software with the Trend reader.)

Figure 2-6. Cable Description - Trend Paper Tape Reader

(The logic diagram of the device cable (below) indicates the connections between the paddle board and the reader connector, P1.)



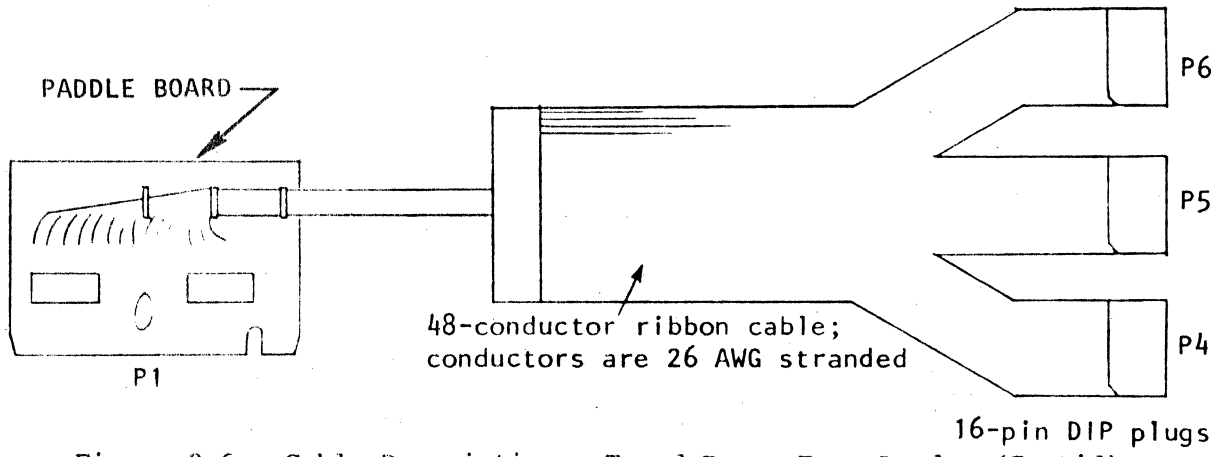




Figure 2-6. Cable Description - Trend Paper Tape Reader (Cont'd)

### 2.1.8 Programming Example

The Assembler Language statements shown in table 2-1 demonstrate one method for using the Distributed I/O System. The technique is based on the information shown in figures 2-4, 2-5 and 2-6 of the Distributed I/O System User's Manual.

The demonstration code falls into three distinct parts: Interrupt Locations, I/O Initiation and End-of-Block Service.

1. Interrupt Locations

   The first part of the code defines and fills the standard locations associated with the IOD channel to which the device's PICOPROCESSOR is connected. As shown in figure 2-4 of the manual, there are six locations involved of which two -- the Byte Count and the Buffer Location -- are dynamic for each I/O Initiation, while the remaining words can be fixed at program load time.

2. I/O Initiation

   The second part of the code accomplishes the transfer of one physical record each time the user's main-line program executes this instruction:

   ```
   CALL PREAD
   ```

   It is assumed that the calling program has previously set up these two words:

   ```
   BYTCNT      Number of bytes to be transferred
   BUFADD      Word address of record buffer
   ```

   The demonstration code converts this information into the form required by an Automatic I/O instruction: negative byte count and byte address of the buffer minus one. Once these computations have been stored into the interrupt locations the PICOPROCESSOR is sent a command specifying:

   ```
   Begin at Branch Address :1
   ```

   Notice that the Device Address used in this sequence is coded with the same Device Address used in the interrupt location, plus one. The assembled object code will indicate the transfer of a command, rather than of data, as shown in figure 2-3 of the manual.

   No attempt is made in the demonstration code to overlap record transfer with CPU activity. Instead, a dead loop is issued -- JMP $ -- until the PICOPROCESSOR interrupts not to the Data Service interrupt location, but to the End-of-Block location. At this point control passes to the subroutine labelled EOB.

3. End-of-Block Service

   The End-of-Block subroutine could perform error analysis, retry, etc. The demonstration code simply obtains device status from the PICOPROCESSOR and passes status back to the main-line program through the A register.

   NOTE

   The programming example can be assembled using either OMEGA or MACRO Assemblers, but must be loaded using either LAMBDA or OS:LNK.

Table 2-1. Programming Example

```
PRINT  EQU    :E8        STANDARD PAPER TAPE READER INTERRUPT
PRDEVA EQU    :FA        STANDARD PAPERTAPE READER DEVICE ADDRESS
*
       ABS    PRINT      INTERRUPT FOR DATA LOCATION
       AIB    PRDEVA     AUTO IN BYTE
       DATA   $-$        TO BE FILLED WITH COUNT
       DATA   $-$        TO BE FILLED WITH BUFFER ADDRESS-1
       DATA   Ø
*                        INTERRUPT FOR END-OF-BLOCK
       JST    *$+1       CALL END-OF-BLOCK ROUTINE
       DATA   EOB
*
PREAD  ENT
       LDA    BYTCNT     # OF BYTES IN RECORD
       NAR               AIB INSTRUCTION NEEDS NEGATIVE
       STA    PRINT+1    PUT IN AIB INSTRUCTION
       LDA    BUFADD     ADDRESS (WORD) OF BUFFER
       LLA    1          AIB INSTRUCTION NEEDS BYTE ADDRESS
       SAI    1          STARTS AT -1
       STA    PRINT+2    PUT IN AIB INSTRUCTION
       LDA    =:Ø21Ø     WORD TO START PICOPROCESSOR
       OTA    PRDEVA+1   SEND TO PICO
       JMP    $          WAIT FOR END-OF-BLOCK
*
EOB    ENT    END-OF-BLOCK INTERRUPT SUBROUTINE
       INA    PRDEVA+1   INPUT STATUS
       RTN    PREAD      RETURN TO CALLER WITH
*                        STATUS IN A REGISTER
```

## 3.1 FACIT PAPER TAPE PUNCH

### 3.1.1 Description

Intelligent Cable 14631-54 controls the transfer of eight-bit positive-true parallel data from the computer to a Facit Model 4070 Paper Tape Punch. The cable employs the same PICOPROCESSOR and software as Intelligent Cable 14631-04 (described in paragraph 3.3.4 of the Distributed I/O System User's Manual) and differs from that cable only in the device connector wiring. The PICOPROCESSOR accepts the IOD output command, interrupts the CPU for data, and then issues a Punch command to perforate the appropriate track of the tape when a "1" is to be entered. When all data has been transferred or when an error is detected, the PICOPROCESSOR generates an end-of-block interrupt to terminate the operation.

### 3.1.2 Physical Details

Cable Lengths:
IOD to PICOPROCESSOR, 4 feet
PICOPROCESSOR to Punch, 1-1/2 feet

Standard Channel Number, 6 (Device Address field = :FC)

Standard Data Service Interrupt Address, :F0

Standard End-of-Block Interrupt Address, :F4

### 3.1.3 Facit Punch Status Word

To control the transfer of data to the paper tape punch, the PICOPROCESSOR sequences through a series of operations based, in part, on the state of individual bits of the Paper Tape Punch Status Word. The Status Word, shown in figure 3-1, indicates certain operational or error conditions in the paper tape punch. When the PICOPROCESSOR receives an input instruction requesting device status, it immediately transfers the entire Status Word to the CPU on bits 0 thru 5 of the data bus. Input status can be requested at any time, but it is usually done after an end-of-block interrupt to determine the reason for termination. Descriptions of the individual status bits are given under 3.1.5.

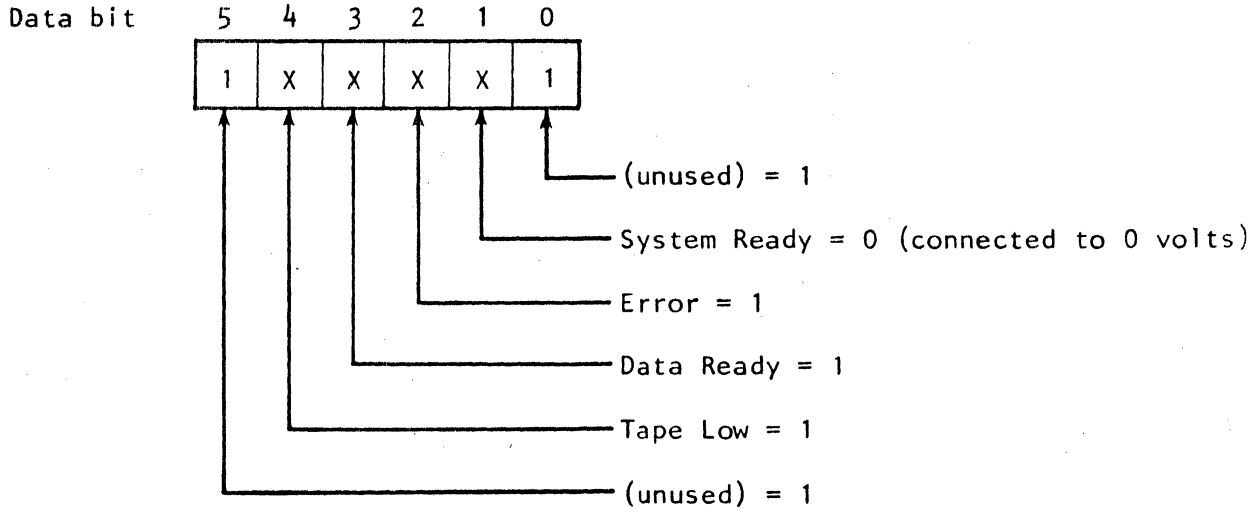



Figure 3-1. Facit Paper Tape Punch Status Word

### 3.1.4 Operating Sequence

The PICOPROCESSOR has 16 unique sequence addresses (:0 thru :F). When the PICOPROCESSOR receives a Command Word with the Begin bit set to "1", it immediately begins operation at the sequence address specified by the Branch Address field of the Command Word. Standard software enters the sequence at the following sequence addresses:

| Sequence Address | Operation |
|---|---|
| :1 | Check Tape Low status; Start Punch Operation. |
| :2 | Start Punch Operation (skip Tape Low status check). |

Details of the PICOPROCESSOR operating sequence are shown in the flow chart (figure 3-2). Interface lines are described in paragraph 3.1.5. Following is a sequence description with each segment of the operation identified by name and sequence address (:0 thru :F). The yes/no decisions refer to the true/false state of a line, regardless of actual logic level.

IDLE (:0)

The PICOPROCESSOR is initially in the Idle state as a result of a Reset command or because of the completion of an end-of-block interrupt. A Begin command with a starting address of :1 or :2 takes the PICOPROCESSOR out of the Idle state and into the Start sequence.

START (:1)

The PICOPROCESSOR begins the Start at the :1 sequence by checking device status. Tape Low is checked first. If this line is true, the PICOPROCESSOR generates an end-of-block interrupt to terminate the operation.

START (:2)

A Start at the :2 sequence begins at this point if it is desired to skip the Tape Low status check. First, error status is checked. The Error status line is true when the tape is broken or loose. On a true response, the PICOPROCESSOR generates an end-of-block interrupt. Otherwise, System Ready status is checked next. Since this signal is connected to 0 volts, the PICOPROCESSOR advances to Data Interrupt.

DATA INTERRUPT (:A)

The PICOPROCESSOR generates a data service interrupt causing the automatic output instruction at the interrupt location to be executed. The CPU transfers data to the PICOPROCESSOR and increments the byte count and memory buffer pointer. A line from the CPU is activated if the byte count increments to zero to signal the PICOPROCESSOR (via the IOD) that all data has been transferred (end-of-block).

While the PICOPROCESSOR checks if the byte count equals 0, the Punch command line is activated. If the byte count (BC) equals 0, the PICOPROCESSOR sustains the Punch command while waiting for the Data Ready line to go false and then, still sustaining Punch, checks System Ready. If System Ready is false, an end-of-block interrupt is generated. Otherwise, data is transferred to the Facit Punch when Data Ready goes true. The PICOPROCESSOR then generates an end-of-block interrupt.

If the byte count is not zero, the PICOPROCESSOR enters the Next Out sequence.

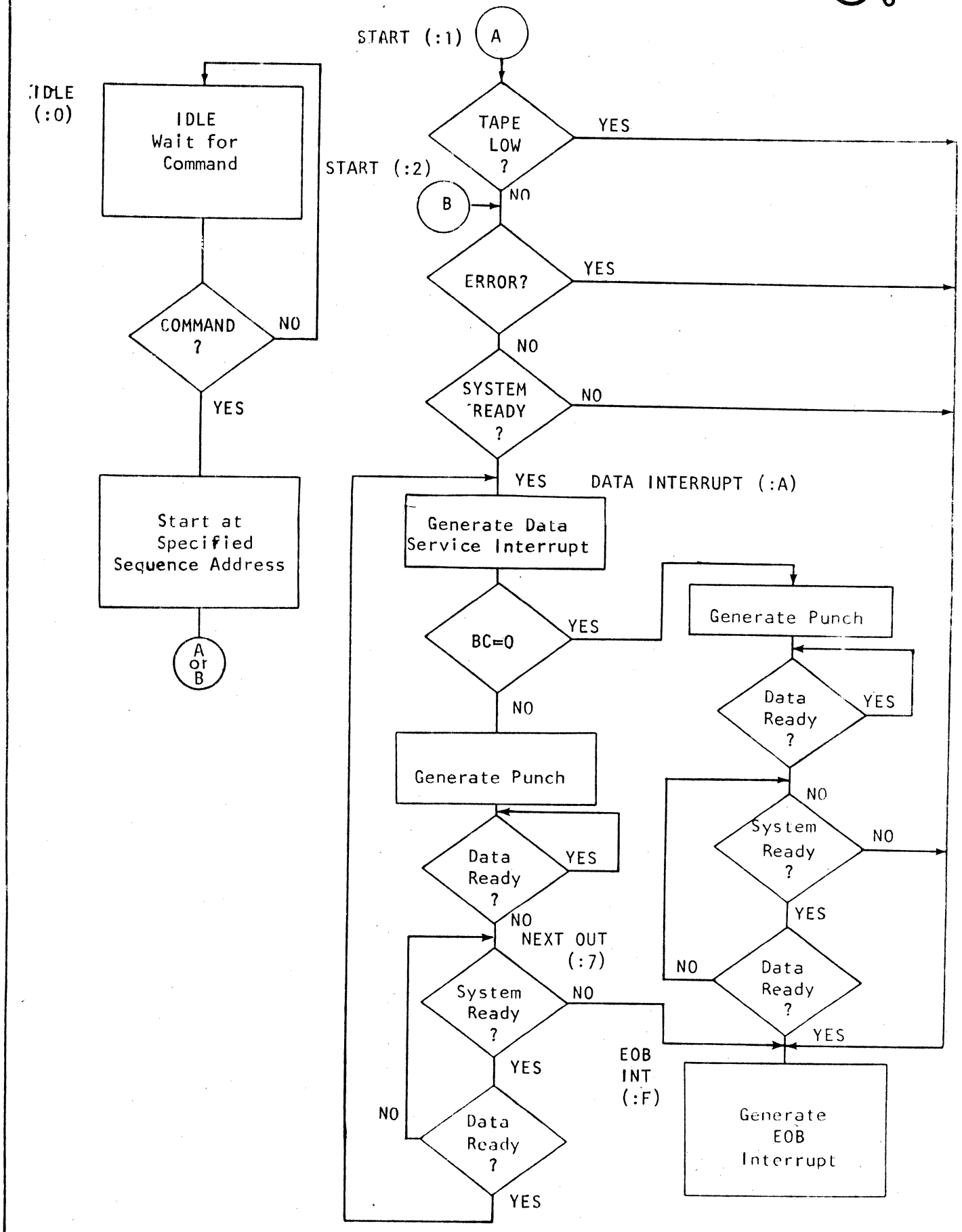



Figure 3-2. Firmware Sequence - Facit Punch Intelligent Cable

NEXT OUT (:7)

The PICOPROCESSOR sustains the Punch command and checks the Data Ready status line. It waits for the Data Ready line to go false, and then checks System Ready, still sustaining the Punch command. Data is transferred to the punch when Data Ready goes true. The PICOPROCESSOR then repeats the Data Interrupt sequence until byte count = 0 is received.

EOB INTERRUPT (:F)

The end-of-block interrupt is generated by the PICOPROCESSOR when the last data byte has been transferred or on detection of a status error at any point in the sequence. When the end-of-block interrupt operation is completed, the CPU commands the PICOPROCESSOR (via the IOD) to return to the Idle state and wait for the next Begin command.

### 3.1.5 Interface Description

Interface lines between the PICOPROCESSOR and the paper tape punch (figure 3-3) include eight data lines, one control line to the punch and four status lines from the punch.

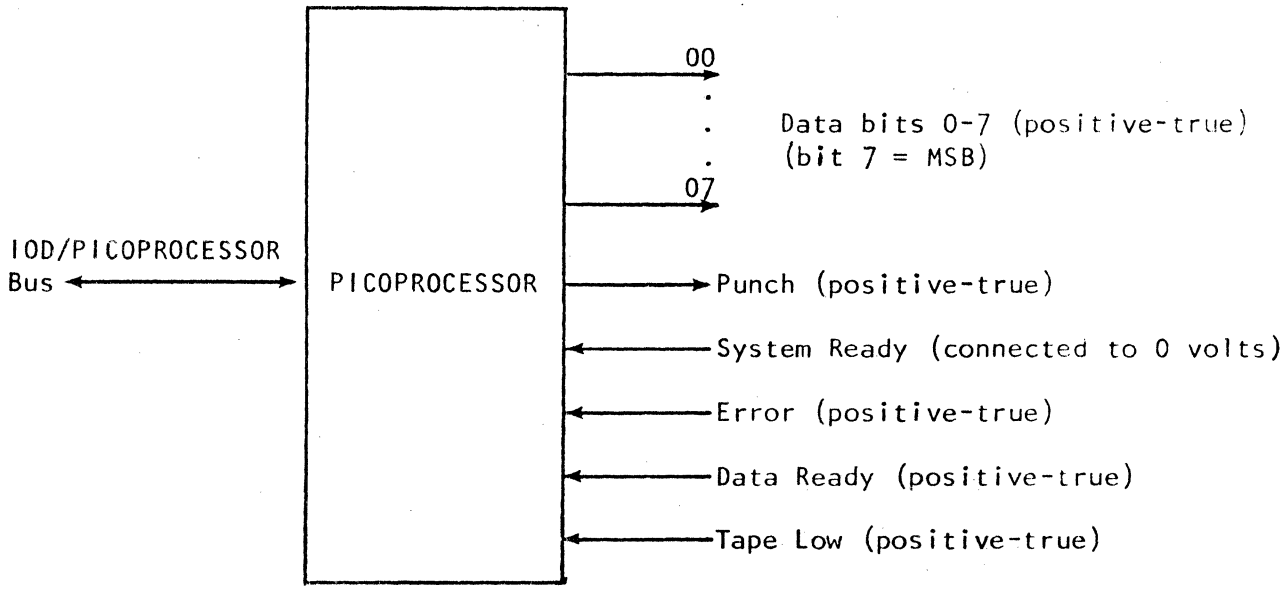



Figure 3-3. Facit Paper Tape Punch Interface

#### 3.1.5.1 Control Lines (to Paper Tape Punch)

1. Punch. This line, when true (+5V), starts the tape moving and initiates punching.

2. Reset. This negative-true line is driven by the CPU RESET switch or under software control. Under software control, it is a 250-ns pulse. This line is not used by the paper tape punch.

3.1.5.2 Status Lines (from Paper Tape Punch)

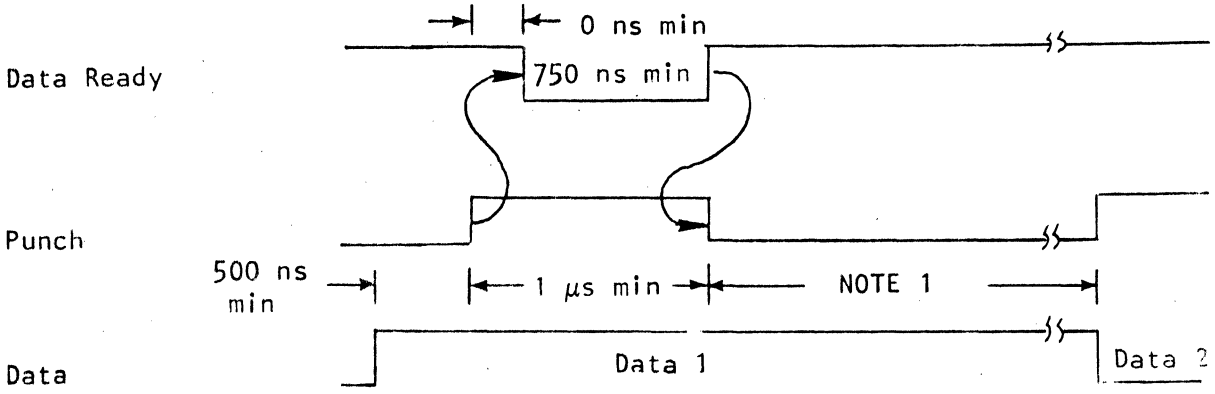
1. System Ready. This line is connected to 0 volts.

2. Error. This line is true (+5V) when the punch is not in the Run mode or when th paper tape is broken, loose, or tight. This line is checked only once for every block of data transferred.

3. Data Ready. This line is true (+5V) when the punch is ready to accept a Punch command. It is false during the "advance and punch" cycle (approximately 13 ms after a Punch command) and when the Error status line is true.

4. Tape Low. This line, when true (+5V), indicates that the tape supply is nearly exhausted. It is for information only. Operation of the punch is not affected. This line is checked once for every block of data transferred, if started at sequence address :1.

Figure 3-4 shows the interface timing. For additional details, see the Facit Paper Tape Punch instruction manual.




NOTE 1: Minimum time equal to the instruction time of the Auto I/O byte instruction plus 4 μs. For instruction time, see the appendix of the appropriate Computer Handbook.

Figure 3-4. Interface Timing - Facit Paper Tape Punch Intelligent Cable

3.1.5.3 Data Lines

The eight data lines to the paper tape punch (0 through 7) are positive-true with bit 7 the most-significant. Data is output to the punch in standard eight-bit ASCII characters.

3.1.6 Strapping Requirements

The Sequence Select factory-installed jumper described in paragraph 4.3.2.2 of the Distributed I/O System User's Manual is installed across pins 5 and 12 of J3 in the PICOPROCESSOR for the paper tape punch. The location of J3 is shown in figure 3-5.

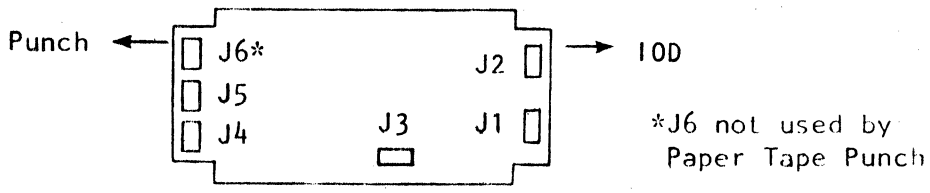



Figure 3-5. Connector Location - Facit Punch PICOPROCESSOR

### 3.1.7 Device Cable Description

The device cable is 18 inches long and terminated on the PICOPROCESSOR end with two 16-pin DIP plugs (P4 and P5). The punch end of the cable is terminated with a 25-pin connector that mates with connector J1 on the paper tape punch.

Figure 3-6 lists all interface lines in the device cable and identifies the connectors used. The location of mating connectors on the PICOPROCESSOR is shown in figure 3-5.

| PICOPROCESSOR | | Description | Punch P1 pin | PICOPROCESSOR | | Description | Punch P1 Pin |
|---|---|---|---|---|---|---|---|
| Conn | Pin | | | Conn | Pin | | |
| P4 | 1 | Data Ready | 12 | P5 | 1 | Not used (input data) | -- |
| | 2 | Error | 20 | | 2 | Ground (not used) | -- |
| | 3 | System Ready (Gnd) | 25 | | 3 | Data bit 06 (Ch 7) | 7 |
| | 4 | Status bit 0 (not used) | -- | | 4 | Data bit 04 (Ch 5) | [illegible] |
| | 5 | Reset (not used) | -- | | 5 | Data bit 02 (Ch 3) | 3 |
| | 6 | Punch | 11 | | 6 | Data bit 00 (Ch 1) | 1 |
| | 7 | Control line 1 (not used) | -- | | 7 | Status bit 5 (not used) | -- |
| | | | | | 8 | Tape Low | 21 |
| | 8 | Control line 0 (not used) | -- | | 9 | Ground (not used) | -- |
| | | | | | 10 | Ground (not used) | -- |
| | 9 | Ground (not used) | -- | | 11 | Data bit 01 (Ch 2) | [illegible] |
| | 10 | Ground (Step Dir.) | 10 | | 12 | Data bit 03 (Ch 4) | 4 |
| | 11 | Ground | 25 | | 13 | Data bit 05 (Ch 6) | [illegible] |
| | 12 | Ground (not used) | -- | | 14 | Data bit 07 (Ch 8) | 8 |
| | 13 | Ground (not used) | -- | | 15 | Ground (not used) | -- |
| | 14 | Ground (not used) | -- | P5 | 16 | Not used (input data) | -- |
| | 15 | Ground (not used) | -- | -- | -- | Ch 9 * | 9 |
| P4 | 16 | Ground (not used) | -- | -- | -- | +6V * | [illegible] |

*Pins 9 and 24 of P1 (punch connector) are tied together.

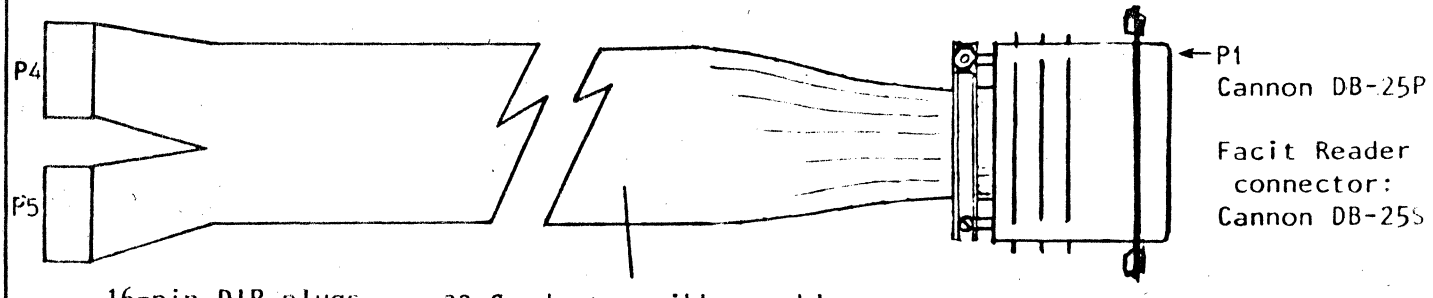



Figure 3-6. Cable Description - Facit Paper Tape Punch

### 3.1.8 Programming Example

The Assembler Language statements shown in table 3-1 demonstrate one method for using the Distributed I/O System. The technique is based on the information shown in figures 2-4, 2-5 and 2-6 of the Distributed I/O System User's Manual.

The demonstration code falls into three distinct parts: Interrupt Locations, I/O Initiation and End-of-Block Service.

1. Interrupt Locations

   The first part of the code defines and fills the standard locations associated with the IOD channel to which the device's PICOPROCESSOR is connected. As shown in figure 2-4 of the manual, there are six locations involved of which two -- the Byte Count and the Buffer Location -- are dynamic for each I/O Initiation, while the remaining words can be fixed at program load time.

2. I/O Initiation

   The second part of the code accomplishes the transfer of one physical record each time the user's main-line program executes this instruction:

   ```
   CALL PUNCH
   ```

   It is assumed that the calling program has previously set up these two words:

   ```
   BYTCNT      Number of bytes to be transferred
   BUFADD      Word address of record buffer
   ```

   The demonstration code converts this information into the form required by an Automatic I/O instruction: negative byte count and byte address of the buffer minus one. Once these computations have been stored into the interrupt locations the PICOPROCESSOR is sent a command specifying:

   ```
   Begin at Branch Address :1
   ```

   Notice that the Device Address used in this sequence is coded with the same Device Address used in the interrupt location, plus one. The assembled object code will indicate the transfer of a command, rather than of data, as shown in figure 2-3 of the manual.

   No attempt is made in the demonstration code to overlap record transfer with CPU activity. Instead, a dead loop is issued -- JMP $ -- until the PICOPROCESSOR interrupts not to the Data Service interrupt location, but to the End-of-Block location. At this point control passes to the subroutine labelled EOB.

3. End-of-Block Service

   The End-of-Block subroutine could perform error analysis, retry, etc. The demonstration code simply obtains device status from the PICOPROCESSOR and passes status back to the main-line program through the A register.

   NOTE

   The programming example can be assembled using either OMEGA or MACRO Assemblers, but must be loaded using either LAMBDA or OS:LNK.

Table 3-1. Programming Example

```
PPINT   EQU     :FØ             STANDARD PUNCH INTERRUPT
PPDEVA  EQU     :FC             STANDARD PUNCH DEVICE ADDRESS
        ABS     PPINT           INTERRUPT FOR DATA LOCATION
        AOB     PPDEVA          AUTO OUTPUT BYTE
        DATA    $-$             TO BE FILLED WITH BYTE COUNT
        DATA    $-$             TO BE FILLED WITH BUFFER ADDRESS -1
        DATA    Ø
*                               INTERRUPT FOR END OF BLOCK
        JST     *$+1            CALL END-OF-BLOCK ROUTINE
        DATA    EOB
*
PUNCH   ENT                     ENTRY POINT FOR PUNCH DRIVER
        LDA     BYTCNT          BYTE COUNT FOR MESSAGE
        NAR                     AOB INSTRUCTION NEEDS NEGATIVE
        STA     PPINT+1         PUT IN AOB INSTRUCTION
        LDA     BUFADD          ADDRESS (WORD) OF BUFFER
        LLA     1               AOB INSTRUCTION NEEDS BYTE ADDRESS
        SAI     1               STARTS AT -1
        STA     PPINT+2         PUT IN AOB INSTRUCTION
        LDA     =:Ø21Ø          WORD TO START PICOPROCESSOR
        OTA     PPDEVA+1        SEND COMMAND TO PICO
        JMP     $               WAIT FOR END-OF-BLOCK
*
EOB     ENT                     END-OF-BLOCK INTERRUPT SUBROUTINE
        INA     PPDEVA+1        INPUT STATUS
        RTN     PUNCH           RETURNS TO CALLER WITH
*                               STATUS IN A REGISTER
```