INTELLEC® SERIES II MICROCOMPUTER DEVELOPMENT SYSTEM HARDWARE INTERFACE MANUAL

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

Order Number: 9800555-03

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

Intel retains the right to make changes to these specifications at any time, without notice. Contact your local sales office to obtain the latest specifications before placing your order.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

The following are trademarks of Intel Corporation and its affiliates and may be used to identify Intel products:

AEDIT	iDIS	Intellink	MICROMAINFRAME
BITBUS	iLBX	iOSP	MULTIBUS
BXP	i_m	iPDS	MULTICHANNEL
COMMputer	iMMX	iRMX	MULTIMODULE
CREDIT	Insite	iSBC	Plug-A-Bubble
i	int_el	iSBX	PROMPT
I^2ICE	int_elBOS	iSDM	Ripplemode
iATC	Intelevision	iSXM	RMX/80
ICE	$int_eligent$ Identifier	Library Manager	RUPI
iCS	$int_eligent$ Programming	MCS	SYSTEM 2000
iDBP	Intellec	Megachassis	UPI

A1005/783/ 4K DD

REV.	REVISION HISTORY	DATE
-01	Original issue.	1979
-02	Manual updated.	1980
-03	Manual updated to support IOC-III. Obsoletes previous editions only for systems with an IOC-III board installed.	7/83

PREFACE

This manual provides detailed information for users who require a comprehensive knowledge of the internal and external interfaces of the Intellec Series II Microcomputer Development System. The information contained in this manual includes definitions of the internal interface commands and their functions, descriptions of the system "firmware" (i.e., ROM-resident programs), and definitions of interprocessor protocol. The intent of this manual is to describe existing interfaces and to explain how these interfaces may be accessed to meet specific user requirements. Readers of this manual are assumed to have a prior knowledge of real-time microcomputer systems and an intimate familiarity with the functional organization of Intellec development system hardware and software. This manual duplicates, in condensed form, information contained in other Intel manuals and also provides a compilation of system and interface information not otherwise available. The manual is divided into the following five chapters:

Chapter 1—Overview. A review of the Intellec Series II development system as it relates to user-accessible interfaces.

Chapter 2—Multibus Interface. A description of the primary user-interface to the development system including ROM and RAM expansion, I/O port assignments, bus priorities and interrupt assignments.

Chapter 3—Serial I/O Interfaces. A description of the two serial I/O channels of the Integrated Processor Board (IPB) or Integrated Processor Card (IPC) and how the channels can be modified to meet user requirements.

Chapter 4—IOC I/O Interfaces. Descriptions of the program interfaces to the integral CRT, keyboard and integral diskette of the I/O Controller (IOC) and of the protocol required for communications between a master processor and the IOC.

Chapter 5—PIO Subsystem Interfaces. Descriptions of the program interfaces to the standard parallel I/O devices (paper tape reader/punch, line printer and PROM programmer) of the Parallel Input/Output (PIO) subsystem and of the protocol required for communications between a master processor and the PIO subsystem.

Appendixes A, B, C and D. Program examples of an Interrupt Routine, Basic Input Driver Routine, Basic Output Driver Routine and Diskette Read/Write Routine.

Appendix E. Interface connector pin assignments and dc signal specifications for the Multibus interface and peripheral interfaces.

The level of descriptions in this manual assume a familiarity with the Intellec Series II development system structure, with the software interfaces of programmable Intel chips, and with the Multibus interface. This prerequisite information can be found in the following Intel manuals:

Intellec Series II CRT and Keyboard Interface Manual, 122029

Intellec Series II Microcomputer Development System Hardware Reference Manual, 9800556

Intellec Series II Model 22X/23X Installation Manual, 9800559

Intellec Series II Microcomputer Development System Schematic Drawings, 9800554

ISIS-II User's Guide, 9800306

Intel Multibus Specification, 9800693

Intel *Component Data Catalog*

Intel *Peripheral Design Handbook*

CONTENTS

TABLES

ILLUSTRATIONS

CHAPTER 1
OVERVIEW

1.1 SYSTEM CAPABILITIES

Intellec Series II Microcomputer Development System, as delivered, is a stand-alone system that can be upgraded by the addition of numerous hardware and software options. Performance of the development system ranges from generation and editing of simple paper tape-based programs to a hard disk-based system capable of supporting assembly or compilation of relocatable library supported code, symbolic debugging of user hardware/software systems, and selective programming of validated user software into PROM. Without modification, the Intellec Series II development system is capable of supporting product development from design inception to the end of the product's life cycle.

There are three basic development system models in the series: The Model 220, the Model 225 and the Model 230. Additionally, there are two variations of each of the basic models according to the operating voltage configuration. For example, a basic Model 220 development system that is configured at the factory for 115 volt operation is designated a Model 220, while a basic Model 220 that is configured for 230 volt operation is designated a Model 221. All models in the series feature an integral video display and an attached keyboard, an integral power supply, a six-slot Multibus-compatible card cage and either one or two flexible disk drives. The development system itself is made up of three microprocessor-based computing elements that are contained on two printed circuit board assemblies. One assembly (either an Integrated Processor Board or an Integrated Processor Card) is inserted into the uppermost slot of the card cage and incorporates the master processor. The other assembly (the Input/Output Controller) is mounted on the inside of the rear panel. This assembly contains the Input/Output Controller (IOC) processor and the Parallel Input/Output (PIO) subsystem processor. Both the Model 220 and the Model 225 include an integral single-density diskette drive and differ only by the circuit board assembly installed in the card cage (the Model 220 uses the 8080-based Integrated Processor Board or "IPB," and the Model 225 uses the 8085-based Integrated Processor Card or "IPC"). The Model 230 is supplied with a separate chassis that contains two double-density diskette drives in lieu of the integral diskette drive common to the Models 220 and 225. The card cage of the Model 230 includes an 8080-based IPB, a 32k RAM board and a two-board double-density diskette controller (to support the two double-density diskette drives).

The high efficiency and cost effectiveness of the Intellec Series II development system, in each of its many configurations are made possible through the use of general-purpose hardware and task-oriented, diskette-based software. This delegation of system personality to software not only simplifies upgrading of development capabilities, but also allows alteration or even replacement of basic system processes. The Intellec Series II development system may thus be viewed as a relatively rigid framework of general-purpose hardware that can be user programmed to meet the needs of a wide variety of applications.

The generation of any resident software requires a knowledge of the system environment within which the software will operate. This manual delineates the Intellec Series II development system environment in terms of the various interfaces between the development system and its subordinate devices and/or external systems.

1.2 MODIFIABLE HARDWARE LOGIC

Although most modifications of Intellec Series II system functions are accomplished by the replacement of software, there are a few hardware changes that may be concurrently necessary. The following text defines changes of this type.

One general rule to be observed when modifying an Intellec Series II development system is that most existing hardware cannot be changed. Aside from reconfiguration of jumpers and replacement of ROM chips, changes to hardware will not only void the system warranty, but may also adversely affect the operation of the supplied software. Furthermore, any hardware rework, including rejumpering or ROM replacement, must meet Intel workmanship standards to maintain warranty provisions. Full warranty rights are preserved only if written permission is obtained from Intel prior to hardware alterations.

Prohibition of hardware changes is not as restrictive as it might first appear. For one thing, the prohibition, whether stated or not, is imposed by the system design which employs buses for communication between intelligent preprogrammed chips. Another factor negating the need for hardware changes is that the Intellec Series II development system hardware was designed in anticipation of other user applications. The Multibus interface and the serial I/O channels are true general-purpose interfaces that can be used for specific applications required by the user.

The following is a list of hardware circuit elements, the use of which may be initiated or altered subsequent to system delivery (usually in conjunction with software changes).

- The serial I/O channels are associated with a number of jumpers that determine the routing and use of data, clock pulses, and control signals for various terminal and data set configurations. The jumpers, located on both the IPB/IPC and IOC boards, are discussed in Chapter 3.

- The 8253 timer of the IPB/IPC uses one counter as a real-time clock that is accessible to user programs executed by the IPB/IPC. The initial count is set via I/O port address F2, and the mode of the 8253 is established via I/O port address F3. On powerup or reset, the real time clock is initialized for a 1ms rate. The clock can be used in conjunction with the local interrupt controller to generate a level 7 system interrupt.

1.3 PROGRAMMING CONSIDERATIONS

A simple description of user programming activities is hindered by two factors. The first of these factors has to do with the large variety of potential applications. Not only does each application dictate a particular minimum of user programming, but in many cases there is a tradeoff between modification of existing software and the creation of new programs or routines. The second factor affecting the user programmer is that Intel-supplied software is physically distributed within RAM and ROM. These two factors make it difficult to detail the software required for a "typical" application without first defining the types of software associated with the Intellec Series II development system. The types of programs to be discussed in the following text are:

- Resident Master Programs — Programs that are executed by the IPB/IPC master processor and provide overall control of the Intellec Series II development system.

- Non-Resident Master Programs — Programs that are executed by another master processor on the Multibus interface and are able to utilize the system resources of the Intellec Series II development system.

- User System Programs — User programs that are being developed to be executed by and to control user-designed hardware systems. Such programs are able to utilize system resources of the development system through the facilities of an In-Circuit Emulator (an Intel supplied non-resident Multibus master).

1.3.1 RESIDENT MASTER PROGRAMS

Resident master programs are usually bootstrap loaded from diskette and executed from system RAM. These programs can also be loaded from paper tape to system RAM or executed directly from ROM. Typical Intel supplied resident master programs are the ROM-based Monitor, the diskette-based ISIS-II diskette operating system, and the diskette-based 8080/8085 assembler. Each of these programs is controlled by specific command entries from the CRT keyboard or system console device that establishes a particular operator interface with the system.

Each of the resident master programs follows the protocol necessary to accomplish data transfer to or from the I/O devices of the development system. However, most Intel-supplied programs employ calls to the Monitor and/or ISIS-II to accomplish I/O transfers. The Monitor provides byte transfers to or from any device (except diskette) whereas ISIS-II accomplishes block transfers to or from any diskette. ISIS-II does not entirely replace the Monitor, but often provides higher-level commands that make use of multiple Monitor calls to simplify operator sequences. The use of Monitor and ISIS-II calls by other programs (including user-designed programs) reduces the program's concern with I/O operations. In effect, the call executes a subroutine within Monitor or ISIS-II that follows the protocol required by the specified device. When the transfer is complete, the calling program continues from the point at which the call was made. Refer to the *ISIS-II System User's Guide* for additional information.

Within the Intellec Series II development system, the protocol for any device involves the use of dedicated I/O port addresses. Furthermore, if the device is subordinate to the IOC or PIO, the protocol requires the issuance of specific commands that control the transfer of data, status, and control bytes between the I/O device controller hardware and the I/O device firmware. Details of the protocol required for each device are discussed in Chapters 3 through 5. A general discussion of protocol is provided in paragraph 1.4.3.

Hardware interrupts may be employed by user programs, but if they are, their use must not conflict with the hardware interrupts of existing circuit boards such as in-circuit emulators. Interrupt level 7 is used by internal circuits within the development system, normally masked off by the Monitor and

ISIS-II since these programs use service requests in lieu of hardware interrupts. Interrupt switches 0 and 1 are used for resetting the Monitor and ISIS-II, respectively.

1.3.2 NON-RESIDENT MASTER PROGRAMS

Non-resident master programs are programs that are executed by Multibus interface master processors other than the IPB/IPC master processor. The external hardware involved must include provisions for interface to the Multibus.

The non-resident master processor has direct access to most of the system resources of an Intellec Series II development system including the Monitor, all of system RAM, and all I/O facilities and devices of the IOC and the PIO. Notable exclusions from the preceding list are the serial I/O channels of the IPB and the IPB's real-time clock and interrupt controllers. These resources on the IPC, with the exception of the system interrupt controller, are available to other bus masters.

The interrupt controllers on the IPB/IPC continue to accept interrupts when another master assumes control of the Multibus interface. The local interrupt controller accepts service requests from external sources (i.e., PIO and IOC) as well as internal sources, and generates a level 7 system interrupt that can be sensed by any master on the bus.

1.3.3 USER SYSTEM PROGRAMS

User system programs are programs that are: 1) assembled or compiled using an Intellec Series II development system; 2) debugged using the combined facilities of the development system and an in-circuit emulator; and 3) programmed into PROM for execution within a user-designed system. In most cases, the physical connection between the user system and the Intellec Series II development system exists only while the user system is being debugged via an in-circuit emulator. However, the in-circuit emulator may also be employed to debug hardware or software that is to directly interface with the Intellec Series II development system.

1.4 I/O DEVICE INTERFACES

Most existing I/O device interfaces are established using hardware and software of the IPB/IPC, the IOC, and the PIO as shown in figure 1-1. Each of the subsystems employs a different combination of hardware and software to accomplish I/O data transfers and to control the subordinate devices. However, the IOC and PIO have several common attributes. The general discussions of I/O device interfaces in the following paragraphs are supported by further details in Chapters 3 through 5.

A second type of I/O device interface is implemented via the Multibus interface. In this case, the hardware controlling the device is not part of the basic Intellec Series II development system and there is some trade-off between the hardware and the driver software executed by the IPB/IPC processor. However, most drivers associated with external I/O device interface hardware make use of Monitor or ISIS-II calls to simplify software design. In any event, any discussion of this type of I/O device interface is more concerned with Multibus interface protocol than with the interprocessor protocol and related topics within this chapter. Refer to Chapter 2 for discussions of I/O device interfaces implemented via the Multibus interface.

1.4.1 FIRMWARE VERSUS SYSTEM SOFTWARE

The term "firmware" is used throughout the computer industry to identify ROM-based microcode that establishes the instruction sets of computers. Within Intellec Series II development systems, the instructions sets of CPUs are fixed, and the term "firmware" identifies ROM-based software. The term "system software" denotes resident and non-resident programs that interpret operator-generated commands. Typical firmware of Intellec Series II development systems includes the I/O device driver programs executed by the IOC and PIO microprocessors.

1.4.2 I/O DEVICE DRIVERS

The complexity of any I/O device firmware is dependent on: 1) the complexity of the hardware interface with the device; and 2) the intelligence of the hardware between the driver and the device. The more complex device interfaces (the serial I/O channels, the integral CRT, and the integral diskette) make use of highly-intelligent programmable chips that tend to reduce driver complexity. However, use of programmable chips imposes a different type of complexity on the associated firmware: the need to include code to initialize the programmable chips during system startup and/or immediately prior to device data transfers. This initialization is required because the operating parameters of the programmable chips must be preselected to meet the specific needs of the system and/or the device being driven.

It would appear that the division of responsibilities among the device driver, the startup routine, and the device interface hardware is highly variable among

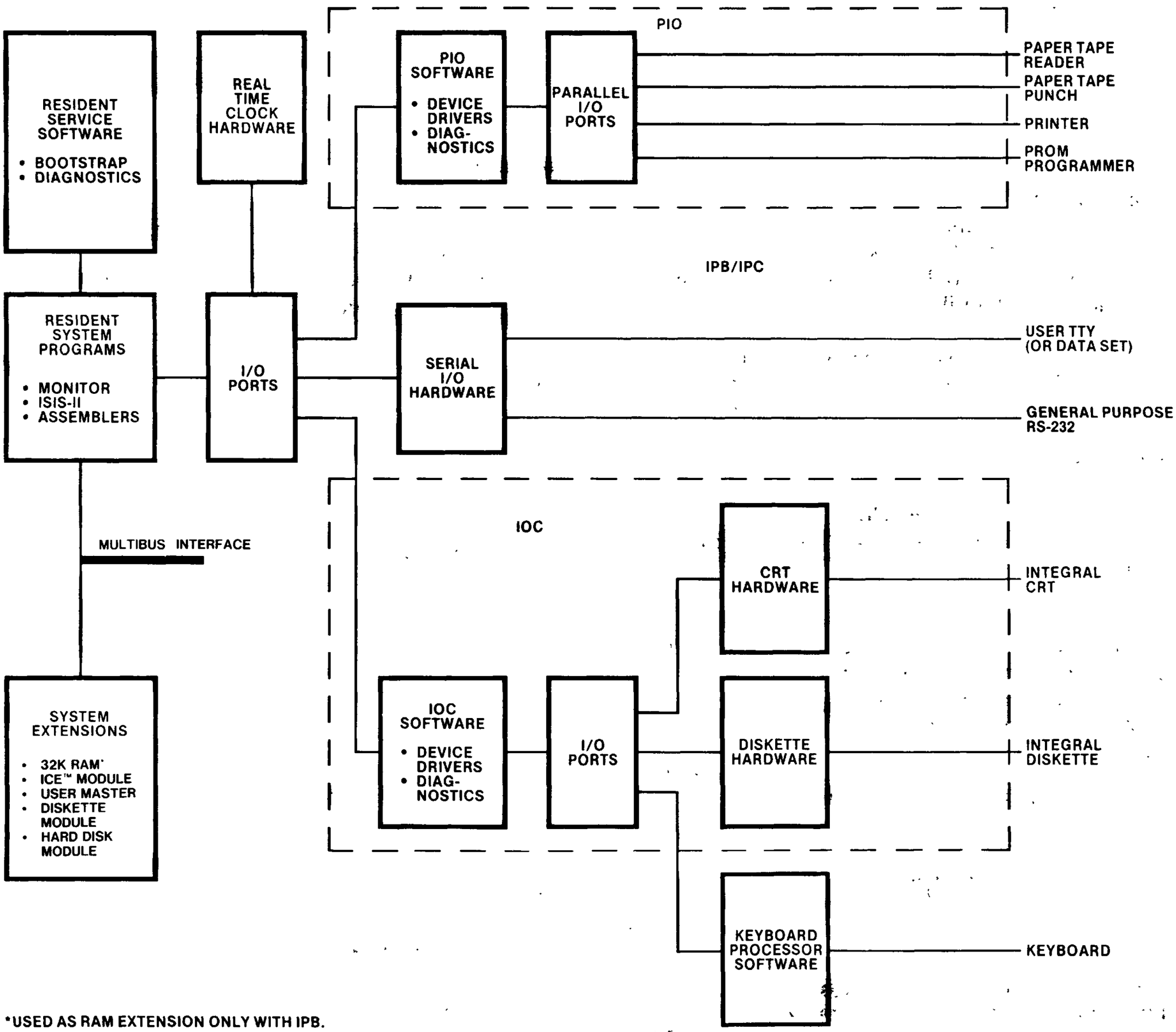


Figure 1-1. Interface Regulating System Elements 555-01

devices. However, one basic criterion employed in the design of the Intellec Series II development system is the maximum simplification of system software I/O processes. The resulting arrangement is such that system software, including ISIS-II, ICE drivers, and user-generated master programs, need only specify the device and either furnish or accept the data to be transferred. Provision is also made to advise the system software when each byte transfer is completed. Some additional complexity occurs with disk transfers in which case the disk controller must be advised of the size of the file to be transferred. In any event, the device drivers have the relatively simple task of passing data bytes between the system software and device interface hardware. The device interface hardware and the startup routines share most of the responsibility for meeting the unique initialization requirements of any I/O device.

1.4.3 DRIVERS/SYSTEM SOFTWARE INTERFACES

There are two types of I/O driver interfaces with the system software. One uses the Monitor interface to accomplish serial I/O transfers. The second type of interface is via the I/O ports of the IPB/IPC master processor and is used by all other I/O device drivers.

The I/O device driver used for serial I/O transfers (i.e., Monitor) is accessed by means of the CALL instruction. The CALL instruction itself merely specifies the proper calling location within the Monitor. Parameters required by the Monitor are placed within the B, C, and (if necessary) the D, E, H, L registers prior to execution of the CALL instruction. System software commands that result in Monitor CALLs often elaborate on the preceding procedure. For example, ISIS-II calls can specify a string of data bytes, and the PL/M statements allow use of the stack to pass additional parameters. However, the resulting process is often a simple reiteration of the CALL instruction.

The second type of driver/system software interface is used to communicate with the IOC and the PIO subsystems of the development system. Each subsystem is accessed via an I/O port address of the IPB/IPC master processor. Each access involves the transfer of a single byte interprocessor command that may be followed by a data byte transfer to or from the subsystem processor. The interprocessor command byte coding informs the subsystem processor if a data byte transfer is to follow and initiates a specific action within the subsystem processor. The command may cause a system level response that affects all subordinate devices of the subsystem or a device level response that affects a specific I/O device and its associated driver and interface hardware.

The system level interprocessor commands initiate actions such as subsystem reset, the return of status concerning the results of a preceding device level command, the enabling or disabling of interrupts, or the return of diagnostic test results. The device level interprocessor commands are used to pass data to or from the device, or to return device status. The sequence of interprocessor commands required to control the subsystem and/or one of its subordinate devices constitutes the interprocessor protocol for that subsystem or device.

1.4.4 TYPES OF DEVICE INTERFACE MODIFICATIONS

The interprocessor command sequence between the IPB/IPC and the IOC or PIO has been generalized to the extent that minor changes of the device/driver interface do not require changes to the protocol. In other words, use of a different printer could require changes to the associated device driver without the necessity for changes to the Monitor, ISIS-II, or any other system software. It is, of course, assumed that the existing hardware interface is compatible with the non-standard printer.

1.4.5 DEVICE SIGNAL TIMING

The interprocessor protocol establishes a relatively loose coupling between the system software of the IPB/IPC and the IOC/PIO device drivers. In general, the system software initiates an I/O operation and then periodically requests I/O status to determine when the operation is completed. This technique relieves the system software of concern with I/O device timing.

The implementation of user-designed I/O drivers and/or resident master programs must take into account the timing of all system elements. For example, hardware interrupts may be employed, but they should be used only to signify the termination of an operation.

High-speed I/O devices, such as disk or diskette drives, require relatively complex hardware and relatively large data storage capacities. Because of this, only one diskette drive is supported by the IOC; the remaining drives employ a disk or diskette controller on the Multibus interface. Furthermore, the integral diskette, the integral CRT, and the refreshing of RAM use a substantial portion of the IOC processor's bandwidth. These factors must be considered when changes to the IOC driver are anticipated.

The lower-speed devices of the PIO make less stringent demands on the PIO processor, but in this case RAM and ROM capacities are more restricted. However, the PROM programmer interface of the PIO is a general-purpose interface that assumes the existence of intelligence external to Intellec Series II. If this intelligence is in the form of a microprocessor, it is possible to establish efficient communications that do not tax the bandwidth or storage limitations of the PIO. Furthermore, such communications can be implemented without changes to the PROM programmer driver. The PROM programmer interface is thus a prime candidate for implementation of a channel to non-standard I/O devices. Refer to Chapter 5 for further information on use of the PROM programmer interface.

The Multibus interface may also be used to communicate with non-standard I/O devices. Refer to Chapter 2 for details on use of the Multibus interface.

1.5 ELECTRICAL CONSIDERATIONS

The Intellec Series II development system, as delivered, is capable of accepting almost any combination of Intel circuit boards. Each development system chassis employs an internal power supply that provides a tie point for circuit grounding and that has adequate reserve power for optional circuit boards. Under normal circumstances, user-designed hardware may be installed either within or attached to the development system chassis without difficulty. However, user-designed hardware must employ compatible grounding techniques and must not exceed the reserve current specification.

1.5.1 GROUNDING

Three types of grounds exist within Intellec Series II development system. The first type of ground is chassis ground that interconnects all metallic enclosures and devices of the system. Chassis ground is routed through the IOC and PIO connectors to provide for grounding of I/O devices. Chassis ground is also used for all cable shields.

The second type of ground is ac ground. This ground is derived from the third (green) wire of the ac power cord. The ac ground is tied to chassis ground at a single point within each power supply.

DANGER

Removal of ac ground from chassis ground can cause hazardous potentials to exist at metallic surfaces of devices and enclosures.

The third type of ground is signal ground. This ground is used as a common reference for all dc voltages and is the ground employed by logic circuits. Signal ground is tied to chassis ground and ac ground at the common tie point within the power supplies.

NOTE

Any Intellec Series II development system incorporating an expansion chassis contains two power supplies, each with its own common tie point for grounding.

User-designed circuit boards are required to use signal ground as a reference for all logic circuits. Chassis ground is used only if the user circuit board is associated with an external enclosure or device (chassis ground is tied to all shielded cables and external metallic structures). Signal and chassis grounds should be isolated on the user circuit board or within the external enclosure or device (if such isolation is possible) to prevent the formation of ground loops.

User-designed systems connected to an Intellec Series II development system via an in-circuit emulator should ideally have independent signal and chassis grounds that may be disconnected from each other when connected to the in-circuit emulator. If user signal ground is permanently tied to user chassis ground, a ground loop will exist. In some cases, this ground loop will cause unwanted currents to flow through the in-circuit emulator signal ground and may result in electrical noise on data, address, and control lines.

Total elimination of ground loops may not be feasible if the system contains peripherals that tie signal ground to chassis ground. When the signal and chassis grounds cannot be separated, a lower-resistance path through the chassis ground wiring should be provided. The installation of heavy (large surface area) straps between the development system chassis and the user system chassis can reduce noise on the signal lines.

1.5.2 POWER SUPPLY RESERVE CURRENT

Two basic power supplies are used in the Intellec Series II development systems: the internal power supply in the development system chassis and a smaller supply in the optional expansion chassis. Both supplies provide regulated voltages of +5, +12, −12 and −10 volts that are available on the backplane. The supply in the development system chassis also provides internal, regulated voltages of +15 and +24 volts for the integral CRT and diskette drive.

The current ratings for each voltage and the amount of reserve current available for optional boards are listed in table 1-1. Note that the expansion chassis contains no circuit boards when delivered, and the current ratings for its four regulated supplies are available for use by optional or user-designed circuit boards installed within the expansion chassis.

Table 1-1. Power Supply Current Ratings

Development System Chassis	+5V	+12V	−12V	−10V	+15V	+24V
Capacity	30.0	2.5	0.3	1.0	1.5	1.7
Model 220 Load	9.7	0.4	0.1	0.02	1.5	1.7
Model 220 Reserve	20.3	2.1	0.2	0.98	0	0
Model 225 Load	10.0	1.5	0.2	0.03	1.5	1.7
Model 225 Reserve	20.0	1.0	0.1	0.97	0	0
Model 230 Load	15.95	0.8	0.1	0.17	1.5	0
Model 230 Reserve	14.05	1.7	0.2	0.83	0	1.7
Expansion Chassis	**+5V**	**+12V**	**−12V**	**−10V**	**+15V**	**+24V**
Reserve	20.0	2.0	0.3	0.8	N/A	N/A

CHAPTER 2
THE MULTIBUS® INTERFACE

The Multibus interface is documented in detail by the *Intel Multibus Specification* and is also described in the *Intellec Series Microcomputer Development System II Hardware Reference Manual*. The information within this chapter does not duplicate the content of these other manuals, but rather defines the Multibus interface characteristics in terms appropriate to Intellec Series II development system users who are writing master programs. In effect, this chapter does not define the Multibus interface as such, but rather describes implementation and/or utilization of system resources via the Multibus interface.

2.1 MEMORY CONFIGURATIONS

The uses of RAM and ROM within a multiprocessor system are varied. Slave processors such as the IOC can use RAM and ROM as local private memory for data and special purpose programs. ROM can also be used to perform logic functions wherein the selection of a given address generates a specified control signal. The above uses do not involve the Multibus interface and do not provide memory that can be shared by other master processors in the system. The following text is concerned only with system memory that is accessible from the Multibus interface.

The size of the Intellec Series II system memory (32k or 64k) is usually more than adequate for most applications. Also, the availability of disk and diskette storage can often reduce the need for additional memory. If, however, the memory size is inadequate, there are means of expanding both ROM and RAM as discussed in the following text.

2.1.1 ROM EXPANSION

When delivered, the Intellec Series II development system contains 4k of ROM. Expansion of ROM is possible through the installation of a Multibus-compatible circuit board using one of two methods. Note that regardless of the method used to expand ROM, the Monitor RAM workspace is always the last (top) 320 bytes of available RAM memory, and care must be taken not to overlay or occupy these locations (refer to the *ISIS-II User's Guide* for detailed information regarding Monitor address spacing). The first method (figure 2-1) simply assigns ROM memory address space. This method reduces the size of the RAM that can be accessed while executing the program out of ROM and is useful when a large ROM-resident program is used to process a relatively small, on-line data base. (The total data base for the system may be very large and stored off-line on diskette.)

NOTE

If several ROM-resident programs are employed, each program should be capable of being separately accessed so that the loss of the corresponding RAM address space during program execution is limited only to the address space occupied by the program.

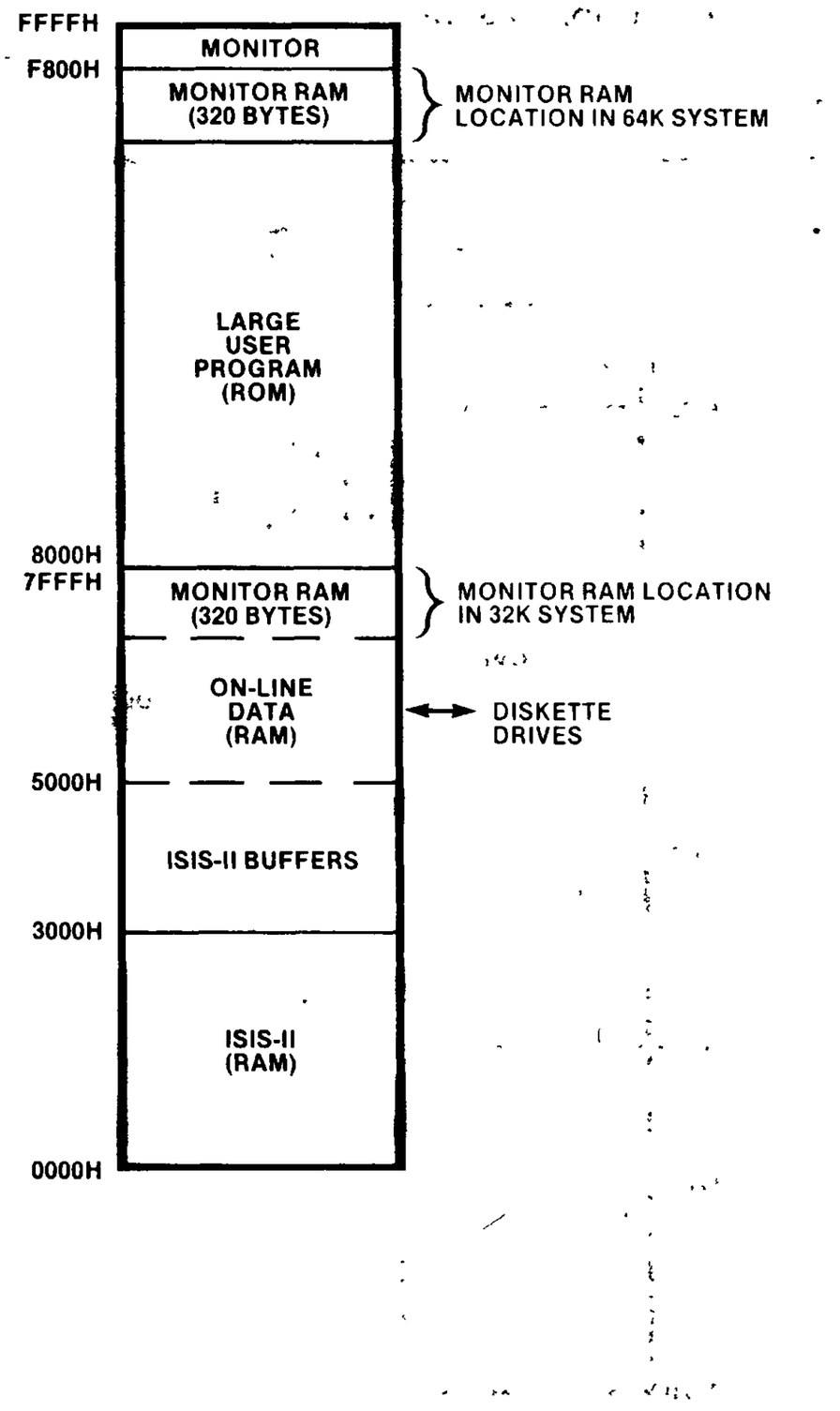


Figure 2-1. Expansion of ROM Address Space

The second method of ROM expansion, known as bank switching (figure 2-2), occupies a limited amount of address space, but allows ROM expansion beyond 64k. With this method, the user-implemented circuit board(s) contains I/O port address decoding logic to select and deselect specific ROM banks. The decoding logic latches the I/O port address and inhibits the selection of more than one memory bank. A typical memory system might contain eight banks of system address space. This system could have a library of relatively-large permanent programs available for immediate execution with minimum reliance on external program storage.

The implementation of either of the preceding ROM expansion methods requires the inclusion of logic that inhibits RAM when the program is being executed out of ROM (INH1) and inhibits ROM when the RAM is being accessed (INH2). The I/O port addresses assigned must be other than those reserved by the IPB/IPC and other circuit boards of the system. For additional information regarding inhibit timing, refer to the *Intel Multibus Specification.*

The bootstrap/diagnostic program is located at addresses E800H through EFFFH and overlays RAM as well as any ROM at these memory locations during initialization and execution of the diagnostic. The selection logic of the IPB/IPC ensures that the initialization routine is executed, without intervention, after start-up or system reset. Comparable techniques (figure 2-3) can be used to execute critical user programs in any memory space except the locations occupied by the Monitor and the bootstrap/diagnostic program. Execution of overlay ROM programs can be initiated via an I/O port address or on exit from the bootstrap/diagnostic program. Use of the ROM overlay technique must not be used simply as a means of ROM expansion, and user hardware would be required to generate INH1 and INH2.

2.1.2 RAM EXPANSION

The 64k RAM available with Intellec Series II development systems is the maximum address space that can be accessed by the master processor of the IPB/IPC. Other master processors on the Multibus interface can employ larger memories wherein the IPB/IPC RAM is but one segment.

The most obvious candidate for a large memory processor is the Intel 8086 microprocessor that uses a 20-bit address to access a full megabyte of memory

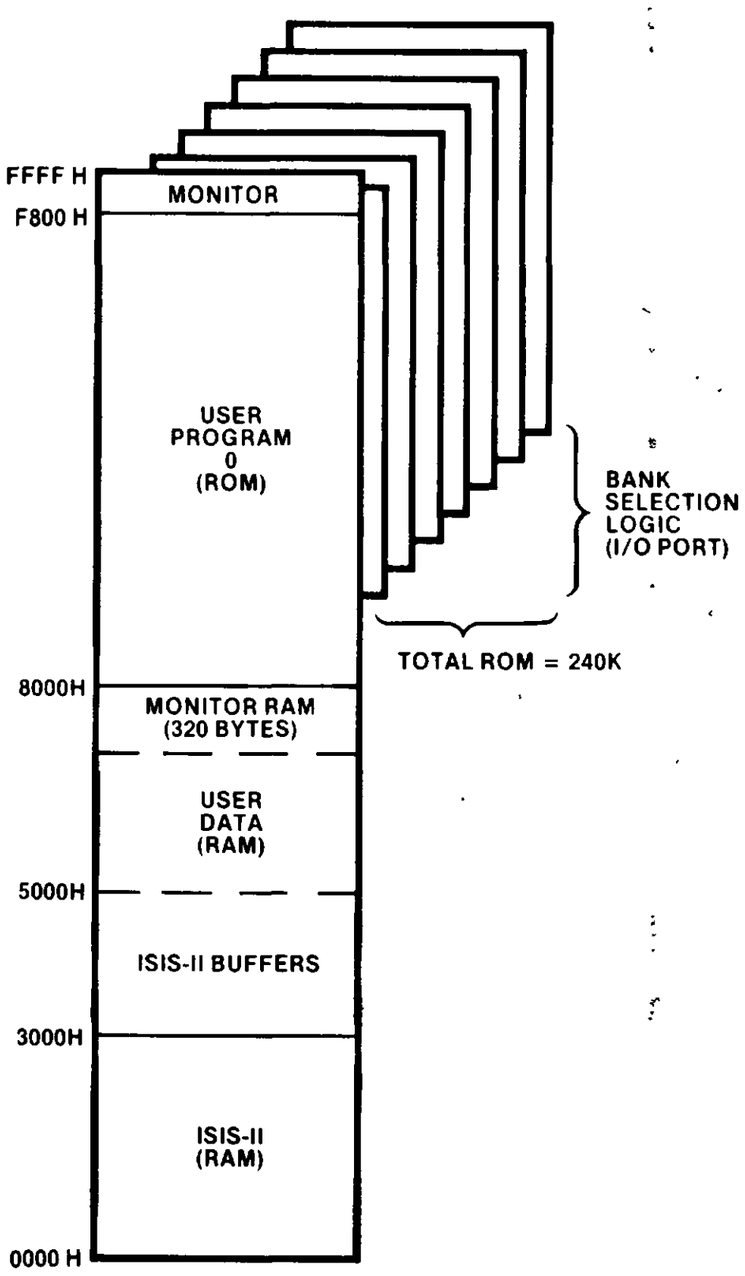


Figure 2-2. ROM Bank Switching 555-03

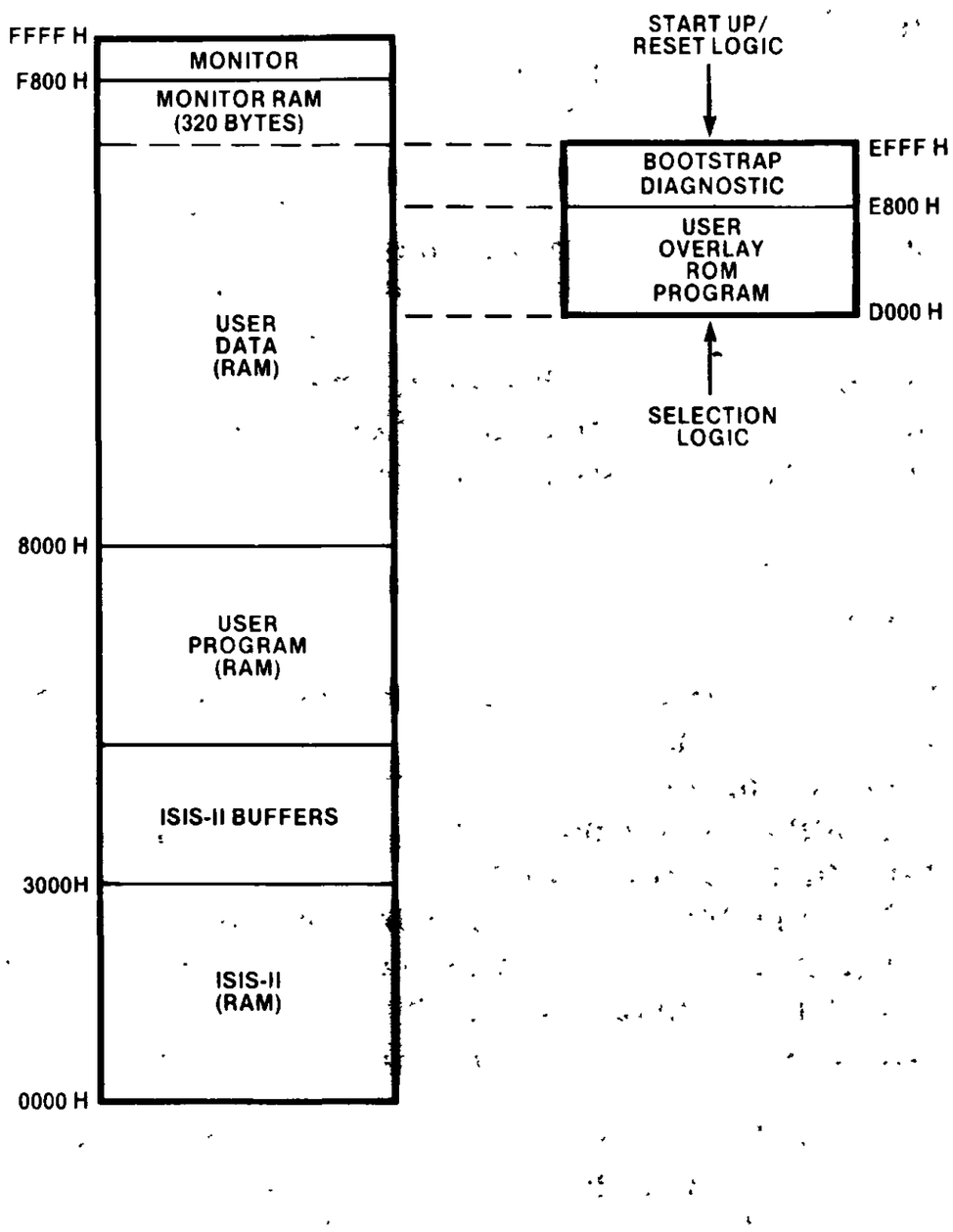


Figure 2-3. Overlay ROM 555-04

(see figure 2-4). The Multibus interface is fully compatible with 8086 system components, but any 8080/8085 type system, including the Intellec Series II development system, must make allowances for the expanded address capabilities of the 8086. With respect to memory addressing, the allowances center on the decoding of the four high-order address bits (ADR10/-ADR13/) of the Multibus interface. Accessing of the RAM segment on the IPB by the 8086 is restricted to byte transfers, while accessing of the IPC's RAM segment by the 8086 does not have this restriction since the RAM on the IPC can be externally accessed in 16-bit words.

The Intellec Series II development systems contain logic on the IPB/IPC that determines when a 20-bit address is being used. This logic assigns addresses 0H through FFFFH to the 64k RAM segment of the IPB/IPC and disables this segment for all addresses greater than 64k. Pull-up resistors associated with the ADR10/-ADR13/ extended address lines are included on the IPB/IPC to ensure access to its RAM segment when an 8086 type processor is not connected to the Multibus interface.

Use of the 8086 is not the only way to expand RAM. User systems that employ 8080/8085 hardware to select memory segments through logical control of the high order address lines can also be used. For example, a user master processor can employ a latched I/O port to set the high-order address lines (see figure 2-5). The user processor can then access either the IPB/IPC RAM segment or other 64k-segments that are inaccessible to the IPB/IPC.

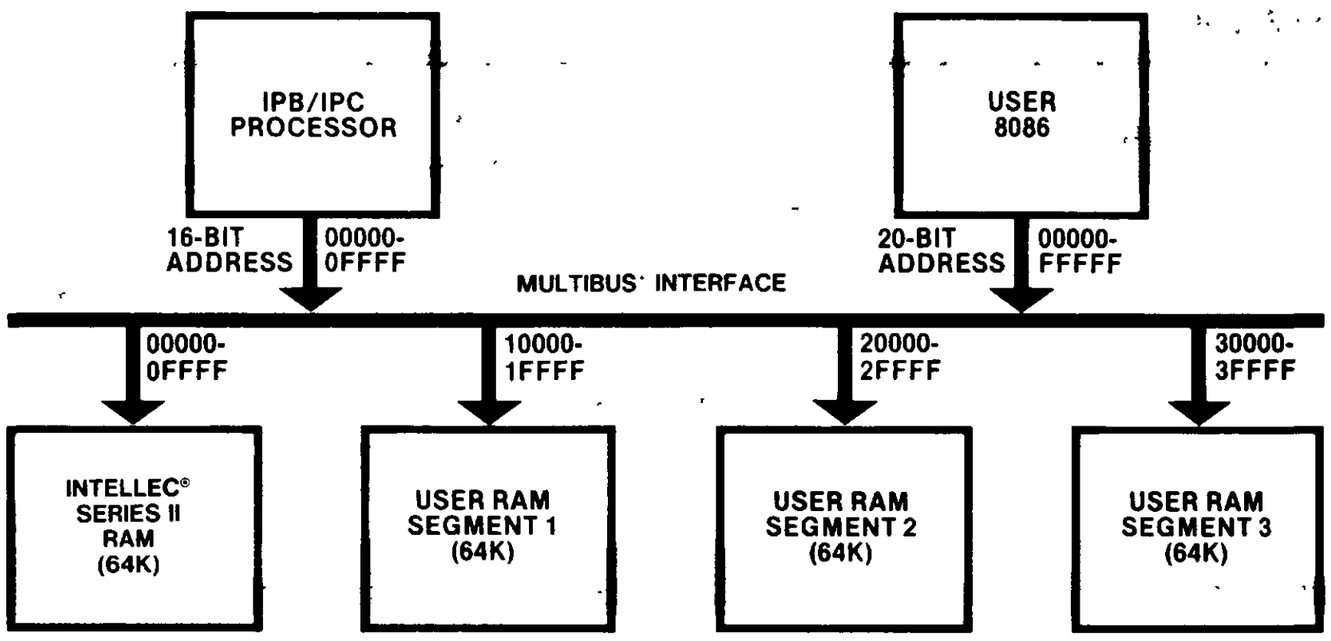


Figure 2-4. Address-Controlled Expanded RAM

555-05

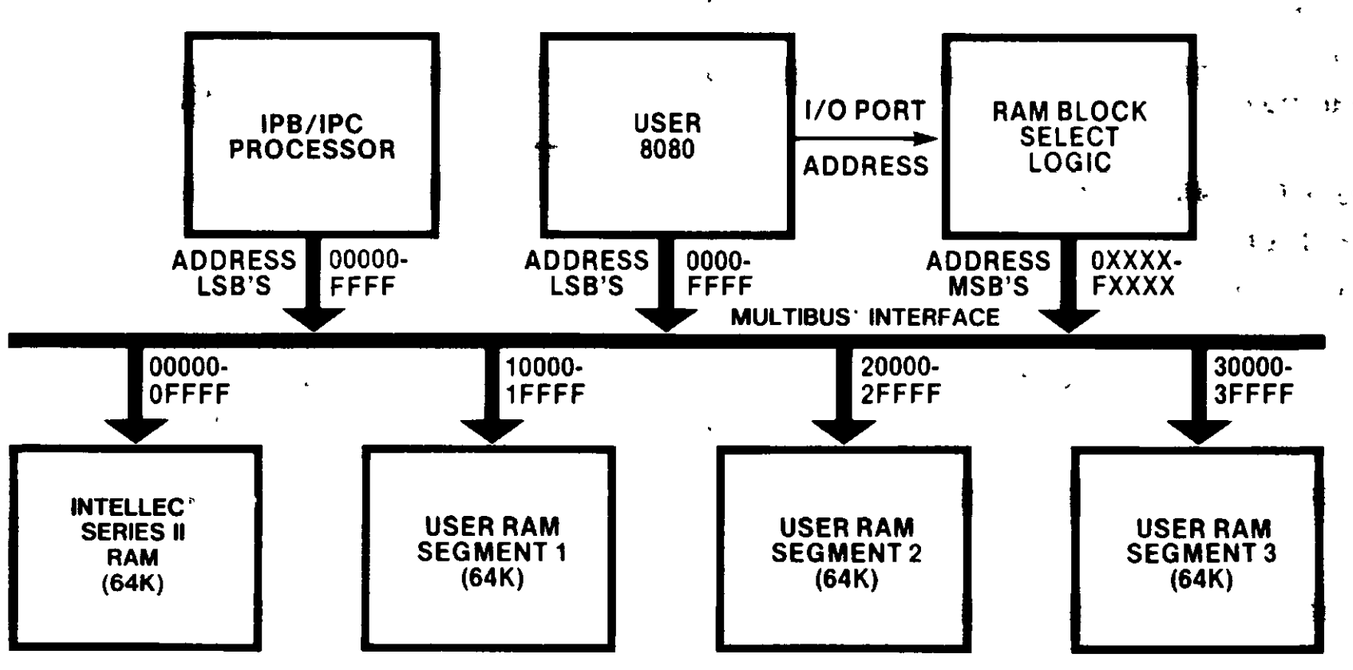


Figure 2-5. Logic-Controlled Expanded RAM

555-06

2.2 I/O DEVICE USAGE

Communications between a processor and its I/O devices are performed similarly to communications between a processor memory in that I/O port addresses access the device or logic associated with the device. These communications become more evident when considering programmable device controller chips with specific registers that are accessed via unique I/O port addresses.

With programmable controllers, the low-order I/O port address bits are used to distinguish between data registers and control functions, and thus a block of I/O port addresses is reserved for each device. The high-order bits are used to select the individual programmable chips. Within the IPB/IPC, IOC and PIO, the I/O port addresses are preassigned and fixed by hardware design. These addresses must be used by user-designed hardware to access the shared system resources.

Table 2-1 lists the I/O port addresses used by IPB, and table 2-2 lists the I/O port addresses used by the IPC. Note that the I/O port addresses are identical for both the IPB and the IPC; only the port accessibility function (shared resource or IPB/IPC accessible-only resource) differs.

A number of Intel products that can be used with the Intellec Series II development systems have dedicated or reserved I/O ports. Table 2-3 lists the current I/O port assignments used by Intel.

Table 2-1. IPB I/O Port Addresses

Shared System Resources	
Port Address	**Function**
C0	IOC data
C1	IOC command and status
C2	Reserved for IOC
C3	Reserved for IOC
F8	PIO data
F9	PIO command and status
IPB Only Resources	
Port Address	**Function**
F0	Serial I/O channel 0 baud rate clock
F1	Serial I/O channel 1 baud rate clock
F2	Real time clock
F3	Timer mode select
F4	Serial I/O channel 0 data
F5	Serial I/O channel 0 command and status
F6	Serial I/O channel 1 data
F7	Serial I/O channel 1 command and status
FA	Local interrupt controller
FB	Local interrupt controller
FC	System interrupt controller
FD	System interrupt controller
FE	Reserved
FF	Control Port

Table 2-2. IPC I/O Port Addresses

Shared System Resources	
Port Address	**Function**
C0	IOC data
C1	IOC command and status
C2	Reserved for IOC
C3	Reserved for IOC
F0	Serial I/O channel 0 baud rate clock
F1	Serial I/O channel 1 baud rate clock
F2	Real time clock
F3	Timer mode select
F4	Serial I/O channel 0 data
F5	Serial I/O channel 0 command and status
F6	Serial I/O channel 1 data
F7	Serial I/O channel 1 command and status
F8	PIO data
F9	PIO command and status
FA	Local interrupt controller
FB	Local interrupt controller
IPC Only Resources	
Port Address	**Function**
FC	System interrupt controller
FD	System interrupt controller
FE	Reserved
FF	Control Port

Table 2-3. Dedicated and Reserved I/O Port Addresses

Device	I/O Port Addresses
ICE-80 In-Circuit Emulator	0E0H-0E3H
Other ICE Modules (ICE-85, ICE-86, ICE-88, etc.)	080H-083H
iSBC 80/05 Single Board Computer	00H-05H
iSBC 80/10A Single Board Computer	0E4H-0EFH
iSBC 80/20-4 Single Board Computer	0D4H-0DFH, 0E4H-0EFH
iSBC 80/30 Single Board Computer	0D8H-0DFH, 0E4H-0EFH
iSBC 86/12A Single Board Computer	0C0H-0CFH, 0D0H-0DFH
iSBC 544 Intelligent Communications Controller	0D0H-0DFH, 0E4H-0EFH
Disks	
First Floppy Diskette Controller	078H-07FH
Second Floppy Diskette Controller	088H-08FH
Hard Disk Controller	068H-06FH

Future Intel products may require I/O port addresses other than the addresses specified in table 2-3. To prevent possible incompatibility with future Intel products, all user-device I/O port addresses should be switch or jumper selectable.

The use of I/O port addresses to access user-designed hardware is the most common technique employed within Intellec Series II development systems. A second technique, called "memory-mapped I/O," may also be employed to access user hardware. With memory-mapped I/O, a block of memory addresses is assigned to the device and/or its controller. The major advantage of memory-mapped I/O is additional programming flexibility; any instruction that references memory can be used to access an I/O port located in the memory space. For example, the MOV (move) instruction can transfer data between any 8080/8085 register and a port or any of the logical instructions can be used to manipulate individual bits within I/O device registers. The simplest implementation of a memory-mapped I/O device is in a system that has unused address space (i.e., a Model 220 with 32k bytes of RAM) in which case the I/O device addresses would be assigned within the 32k to 62k address range. However, in systems containing 64k of RAM (i.e., the Models 225 and 230), the I/O device addresses must replace either RAM or ROM. RAM replacement is impractical as it would necessitate the physical removal of RAM. ROM replacement requires I/O device control logic that generates INH1/ (to inhibit RAM) when the I/O device is being used. Since most memory-mapped I/O devices require very few addresses, ROM replacement is the most widely used method for memory-mapped I/O. However, programmers must be aware that the addresses assigned to an I/O device are no longer accessible in RAM.

2.3 INTERRUPT MECHANISMS

The standard definition of an interrupt is the process whereby an asynchronous device-generated signal causes program branching to a routine that services the needs of the device (usually the handling of I/O data). However, the reason for employing interrupts is that I/O devices (or other real-time system elements) often involve synchronous operations that cannot be delayed while the processor is performing a lengthy processing task. An interrupt service routine is used to quickly complete the I/O data transfer and then allow continuation of the interrupted process. At a later time, the program processes the data accepted by the service routine or prepares new data for output to the device. Without interrupts, input data could be lost or output data could be unavailable when required.

There are two reasons why the preceding real-time hardware interrupts are not required when an Intellec Series II development system is executing many of the Intel-supplied resident master programs (Monitor, ISIS, etc.). The first reason is that I/O operations are program controlled to be performed in sequence (e.g., if diskette data is to be printed, the diskette operations are completed before printing

operations begin). The second reason is that interactions with high-speed I/O devices are provided by means of controller chips that perform many of the tasks (including temporary data storage) that would otherwise be performed by interrupt service routines. The master program is never concerned with multitasking or with close synchronization of I/O device operations. Note that while some input (e.g., keyboard entry), is not program controlled, its data entry rate is slow enough to allow temporary data storage by hardware while the processor is occupied.

In place of hardware interrupts, the hardware elements of the Intellec Series II development system use service requests (specified bits within status bytes) to determine when a device requires attention. The status bytes are returned to the master processor of the IPB/IPC on demand. The transfer of status bytes from the IOC or the PIO to the IPB/IPC constitutes a major segment of the interprocessor traffic. Although hardware interrupts are not widely used by Intel-supplied resident master programs, interrupt handling circuits exist within the Intellec Series II development system. These circuits establish priorities for the interrupt switches, for interrupts from the Multibus interface and for interrupts from internal hardware elements of the IPB/IPC, IOC, and PIO. The interrupt circuits also apply interrupt switch and internal hardware interrupts to the Multibus interface. Interrupt masking by resident programs inhibits sensing of interrupts by the IPB/IPC master processor. All interrupts generated by internal hardware elements are handled by a local interrupt controller that operates in the polled mode as a slave to the system interrupt controller. All internal interrupts are processed by the local interrupt controller and generate a level 7 interrupt to the system controller. Resident master programs must then poll the local interrupt controller to determine the source of the internal interrupt. An example of an interrupt routine used to service an interrupt originating from a device associated with the local interrupt controller is shown in Appendix A. Interrupt level assignments for the local interrupt controller are as follows (level 0 has the highest priority):

Level	Function
0	Serial I/O Channel 0 Input Data Ready
1	Serial I/O Channel 0 Output Data Ready
2	Serial I/O Channel 1 Input Data Ready
3	Serial I/O Channel 1 Output Data Ready
4	1ms Real Time Clock Interrupt
5	PIO Subsystem Interrupt
6	IOC Interrupt
7	Not Used

The system interrupt controller operates in the fully-nested mode and is initialized with a call address interval of eight and a base address of 0H to establish the location of the vector address block. The vector addresses reserved for system interrupts are as follows:

Interrupt Level	Vector Address	Vector Usage
0	00H	Monitor
1	08H	ISIS-II
2	10H	Disk Controller
3	18H	
4	20H	ICE-80 Module
5	28H	
6	30H	ICE Modules
7	38H	Local Interrupt Controller

CAUTION

Reprogramming the 8259's call address interval from eight to four will cause undefined system operation.

The local and system interrupt controllers of the IPB and the system interrupt controller of the IPC cannot be programmed or polled by a non-resident master program (the local interrupt controller of the IPC can be accessed by another bus master). When another bus master assumes control of the bus, both the local and system interrupt controllers maintain any current interrupt request and latch any subsequent interrupt request (when the IPB/IPC regains bus access, any pending interrupt request is serviced). Since all system interrupts can be sensed by another bus master via the Multibus interface, all IPB/IPC local interrupts can also be sensed (any local interrupt causes a level 7 system interrupt).

2.4 MULTIBUS® PRIORITY LOGIC

To avoid conflicts that may arise when two or more bus masters simultaneously require bus access, the IPB/IPC includes parallel priority resolution logic. This logic accepts individual bus request inputs from up to nine bus masters that may be installed in the backplane (five available slots in the development system chassis and four slots in the expansion chassis) and returns an individual bus priority input to all but the bottom slot of the expansion chassis (the bottom slot has the highest priority and its bus priority input is permanently enabled). The parallel priority logic samples all of the bus request (BREQ) inputs and generates an individual bus priority in (BPRN) output to the highest priority bus master requesting the bus. Following Multibus interface protocol, when the requesting master receives its bus priority in signal, it examines the common bus busy (BUSY) bidirectional line to determine when the bus becomes available and, when the bus is available, activates bus busy to indicate to all other bus masters

that the bus is in use. Note that a bus master maintains bus access until it either releases the bus (i.e., the bus master executes a halt instruction) or until a higher-priority bus master requests the bus.

Bus priority within the Intellec development system (including the expansion chassis) is assigned in a bottom-up sequence with the top slot (the IPB/IPC) having the lowest priority and the bottom (tenth) slot having the highest priority. The use of parallel priority logic (rather than serial priority logic) allows the priority of any bus master (except the IPB/IPC) to be readily changed by relocating the board in the backplane and allows "slave" boards (boards that do not request the bus) to be positioned anywhere in the backplane without affecting bus priority. When positioning bus masters in the backplane, boards that have high-speed transfer rates or high data volume functions (e.g., disk controllers) should be placed in the lower (higher priority) slots, and low-speed device controllers should be placed in the higher (lower priority) slots.

2.5 REAL-TIME PROCESSING

Prior discussions in subsection 2.3 describe how an Intellec Series II development system performs its tasks in serial fashion so that the IPB/IPC master program devotes its full attention to the current operation. For I/O devices, this sequential processing approach is more than adequate since the I/O operation appears to the operator to be immediately performed. However, when an Intellec Series II development system is employed in a real-time environment, the techniques used with the I/O devices may or may not suffice. The following paragraphs define the capabilities and limitations of real-time techniques that may be employed by the user.

2.5.1 USE OF INTERRUPTS

Anyone having prior familiarity with real-time, interrupt-driven systems might be prone to design a master program that makes use of asynchronous interrupt requests in place of the service requests that are currently used by Intel-supplied software. This design would make the system more responsive to real-time events. However, the user-designed program must employ interrupt service routines with execution speeds that do not interfere with normal operation of high-speed I/O devices. Furthermore, drastic program alterations, such as the implementation of concurrent I/O operations, would be ill-advised because of the impact that the alteration might have on existing operating systems, I/O drivers, and I/O interface hardware. The alternative methods discussed in the following paragraphs should be seriously considered before attempting extensive use of real-time interrupts.

2.5.2 USE OF SLAVE PROCESSORS

In some cases, the occurrence of an external real-time event requires the immediate initiation of a relatively long program sequence, but does not require immediate interaction with other elements of the system. In such cases, the task may be assigned to a user-designed slave processor subsystem.

Two examples of slave processor subsystems are the IOC and PIO. The IOC and PIO are functionally limited in that they perform most of their operations only in response to commands from the IPB/IPC (CRT and RAM refresh within the IOC are continuous, but are implemented by special-purpose hardware). User-designed slave processor subsystems can use software that continually interacts with the external interface and can use interrupts to signify when the slave processor requires service from the master processor.

Implementation of a user-designed slave processor subsystem can be accomplished without physical alteration of the Intellec Series II development system. (The slave processor would contain I/O port address decoding logic that monitors the address lines of the Multibus interface; the I/O port addresses employed must be among those not reserved by Intellec Series II development system functions.) The slave processor subsystem is, in some respects, equivalent to a hardware-implemented interrupt service routine in that it allows asynchronism between the control of real-time operations and the timing of IPB/IPC processing. However, the slave processor has the distinct advantage of performing complex operations without the necessity of borrowing time from other IPB/IPC operations. A slave processor tends to increase the bandwidth of the entire system. The services of the slave processor are equally available to the IPB/IPC and any other master processor on the Multibus interface. Note that with any user-designed slave processor, some form of synchronization protocol must be established among all master processors that use the slave processor in order to prevent concurrent access of the slave by more than one master.

2.5.3 PARALLEL PROCESSORS

The use of parallel processors substantially increases system bandwidth by permitting two or more master processors to simultaneously execute system level programs.

The 8086 microprocessor is ideal for parallel processor configurations since its architecture directly supports the interfacing of two independent buses (any microprocessor can drive two buses). A typical arrangement within an Intellec Series II development system might use one of the buses to communicate

with the IPB/IPC that would be used to control I/O operations. The second bus would then be used to execute master programs residing in unshared RAM or ROM. Both buses could be the Multibus interface or two types of buses could be implemented in much the same way as the IOC processor interfaces with both the Multibus interface and its own IOC bus. The number of possible implementations is limited only by the imagination of the designer.

Major decisions in the design of a parallel processor system are the allocation of tasks to the processors and the establishment of interprocessor communications. These two decision areas are interrelated in that the division of responsibilities between the processors determines the amount of data and status that is exchanged. If large amounts of data must be exchanged, the two processors could use a shared RAM that the user processor accesses via the Multibus interface. If smaller amounts of data are exchanged, a hardware-implemented data bus buffer (DBB) could be used (refer to IOC description within the *Intellec Series II Microcomputer Development System Hardware Reference Manual*). In using a DBB, the user processor appears to be a slave processor to the IPB/IPC. Interrupts can be employed, and the status bytes returned by the user processor can redirect IPB/IPC operations and thereby assume control of the system. The use of a DBB is somewhat limited by its low data-transfer rate.

The preceding information is provided only to outline some of the possible methods of implementing parallel processor configurations, and many details are omitted. Nevertheless, parallel processing can be implemented without modification of Intellec Series II development system hardware or firmware.

CHAPTER 3
SERIAL I/O INTERFACES

siehe auch S. 2-4
Port adresses

The two serial I/O channels are the only I/O interfaces of the Intellec Series II development system that can be directly accessed by the IPB/IPC. All logic associated with the serial I/O channels is incorporated on the IPB/IPC or within its associated software and firmware. The connection between the serial I/O channels and the serial device connectors on the rear panel is accomplished by etched traces on the IOC board assembly (the IOC includes a number of alterable jumper links that are used to configure connector signal routing).

The direct path between the IPB/IPC and the serial I/O devices has the advantage that in terms of overall operation, the omission of an intervening slave processor (i.e., the IOC or PIO) can make the system more responsive to serial I/O events. However, if interrupts are masked-off or disabled (normal configuration), the response time of the system to any serial I/O input is still dependent on the frequency at which the IPB/IPC software polls the serial I/O interface. Polling is continually performed since most serial I/O devices (teletypewriters, video terminals and modems) provide input that cannot be anticipated by local programming. The polling rate employed is sufficient to support synchronous communications at rates of up to 64k baud.

3.1 SOFTWARE ALTERATIONS

The Intel-supplied software associated with the serial I/O channels is part of the system software (ROM-resident Monitor and bootstrap/diagnostic programs). ISIS-II has the ability to indirectly initiate data transfers via a serial I/O channel (ISIS uses the facilities of the Monitor to accomplish I/O transfers). With the IPB, all other system software must use Monitor (or ISIS) calls to access the serial I/O channels. With the IPC, the serial I/O channels are defined as system resources and can be directly accessed by another bus master.

The Monitor can only access a serial I/O channel when the channel is defined as the system console. By default, the integral CRT and keyboard are defined as the console device when the system is initialized, and the I/O ports associated with the serial I/O channels are unused. Note that once the system has been initialized, the Monitor's A (assign) command can be used to assign a device connected to one of the serial I/O ports as the console device. Access to a serial I/O channel that is not defined as the system console is possible only through a user-designed I/O driver.

The bootstrap/diagnostic program initializes the serial I/O channel hardware during start-up and following system reset. The initialization sequence is simply the transfer of operating parameters to the 8251 universal synchronous/asynchronous receiver transmitters (USARTs) and the 8253 programmable interval timer. Specifically, both the channel 0 and channel 1 USARTs are initialized for asynchronous operation with two stop bits, an 8-bit character length and a baud rate factor of 16X. The 8253 interval timer consists of three separate counters that are used to determine both the baud rate clock frequencies for the USARTs and the real time clock frequency. All three counters are initialized for Mode 3 (square wave) operation with a two-byte count register. The counter register values provided to each counter when the timer is initialized are as follows:

Counter	Function	Counter Value	Counter Frequency
0	Channel 0 Clock	698	1.74 kHz
1	Channel 1 Clock	32	38.4 kHz
2	1 ms Real Time Clock	1229	1 kHz

Note that the channel 0 and channel 1 clock signals are subsequently divided by 16 (baud rate factor 16X) by the USARTs to provide baud rates of 110 and 2400 baud, respectively.

Alteration of the ROM-based bootstrap/diagnostic program to change the baud rate or operating mode is not necessary since the serial I/O channel hardware (the 8251s and the 8253) can be reinitialized by a user-designed initialization routine that is executed following the bootstrap/diagnostic program. An example of a routine that modifies the I/O channel baud rate is provided in Appendix B of the *Intellec Series II Model 22X/23X Installation Manual*. For asynchronous I/O devices, the modifiable parameters include character length, the number of stop bits, parity enable/disable and odd/even parity selection as well as baud rate. For synchronous I/O devices, the modifiable parameters include character length, parity enable/disable, odd/even parity selection, sync in/out and single/double sync character selection. (Refer to the Intel *Peripheral Design Handbook* for details on programming the 8251 USART.) Note that when reprogramming the 8251, the mode instruction is recognized only after the 8251 has been reset (external rest or 8251 command instruction with internal reset bit set). Since the command instruction is used to define the states of the serial I/O channel control lines, this instruction is normally output prior

to the transfer of serial I/O data. If the internal reset bit is inadvertently set or if a mode instruction does not immediately follow each use of the internal reset bit, the serial I/O channel will be inoperative until it is again initialized.

3.2 HARDWARE ALTERATIONS

Hardware alterations associated with the serial I/O channels are limited to changes of jumpers on the IPB/IPC and the IOC circuit boards. This limitation is imposed not only to maintain functional integrity of the Intel-supplied circuit boards, but also because the circuit board designs are dictated by requirements of the 8251 and 8253 chips.

The 8251 chip has two control lines (RTS and DTR) that are, in reality, program-controlled general-purpose output signals. Similarly, DSR is a general-purpose input signal that can be used to signify a variety of conditions at the external device. With appropriate programming, the serial I/O channels are compatible with many types of serial devices and, since the RS-232 interface is used by most data sets, the serial device can be remote from the development system site (i.e., telephone lines can be used). Also, with the possible addition of some external hardware, the serial I/O channels can be compatible with any TWX or TELEX network.

The jumpers on the IOC circuit board are used to interconnect or crossover signals to and from the serial I/O channels on the IPB/IPC to conform to the pin assignments of the I/O device to be interfaced. All of the functions performed by the IOC jumpers can be implemented within the I/O device cabling; the jumpers eliminate the need to rewire the device cable/connector. To determine the jumper positions required for any I/O device, refer to Appendix A of the *Intellec Series II Models 22X/23X Installation Manual* and match the pin assignments of the I/O device with the rear panel connector (J2 and J3) pin assignments. Note that when interfacing a serial I/O device to the SERIAL CH 2 connector (J3), the transmit and receive data lines may have to be reversed (remove W7 jumpers A-B and C-D and install W7 jumpers A-C and B-D). Also, if the serial I/O device does not generate CTS (clear to send), remove W1 jumpers A-B and C-D and install a single W1 jumper at A-C to connect the RTS (request to send) output to the CTS input.

The jumpers located on the IPB/IPC circuit board are used to establish the source of the receive and transmit clock signals for both serial I/O channels. The jumpers installed at the factory configure both channels for asynchronous operation by routing the CH0 CLK and CH1 CLK signals from the 8253 programmable interval timer to the receive clock (RXC) and transmit clock (TXC) inputs of the USARTs. For synchronous communications, the jumpers must be repositioned so that the RXC and TXC signals originate from the serial I/O device through the serial I/O connector. Table 3-1 defines the jumper positions for both asynchronous (factory installed) and synchronous communications.

3.3 INTER-SYSTEM COMMUNICATIONS

It is sometimes advantageous to couple two Intellec Series II development systems together to allow sharing of system resources or to meet the need for intersystem communications. If the two systems are remote from one another, the use of data sets is necessary. However, if the two systems are in close proximity, a pseudo data set can be implemented in the form of an interconnecting cable. Jumpers within each system are configured so that the clock for synchronous communications is derived from the transmitting system.

Figure 3-1 shows the jumpering and cable wiring necessary for intersystem communications. The data-set simulator cable shown is for serial I/O channel 0 (SERIAL CH 1 connector) of both systems. In this case, the external transmit clock is routed via pin 24 of connector J2. (When current loops are employed, pin 24 is used as a return for the receive data signal RXD RETURN.) The illustrated arrangement can be used for serial I/O channel 1 (SERIAL CH 2 connector), but in this case the jumper for the external TX clock is located on the IOC circuit board.

Table 3-1. Asynchronous/Synchronous Jumper Configurations

Asynchronous Operation		Synchronous Operation		Function
Channel 0	Channel 1	Channel 0	Channel 1	
14-15	1-2	13-14	1-3	Transmit Clock
11-12	4-5	10-11	5-6	Receive Clock
7-8*	N/A	8-9	A-B**	External Transmit Clock

*TTY Receive Data Return

**Install jumper W3 (A-B) on IOC circuit board.

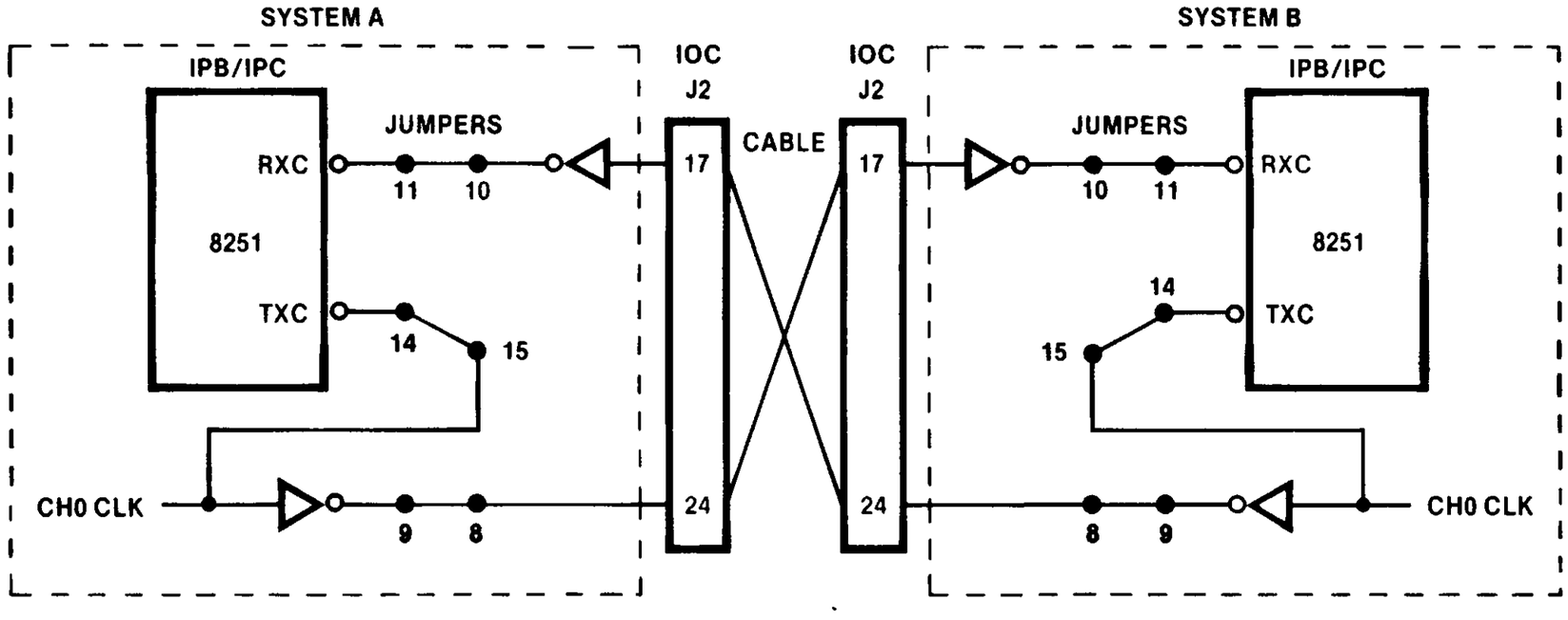
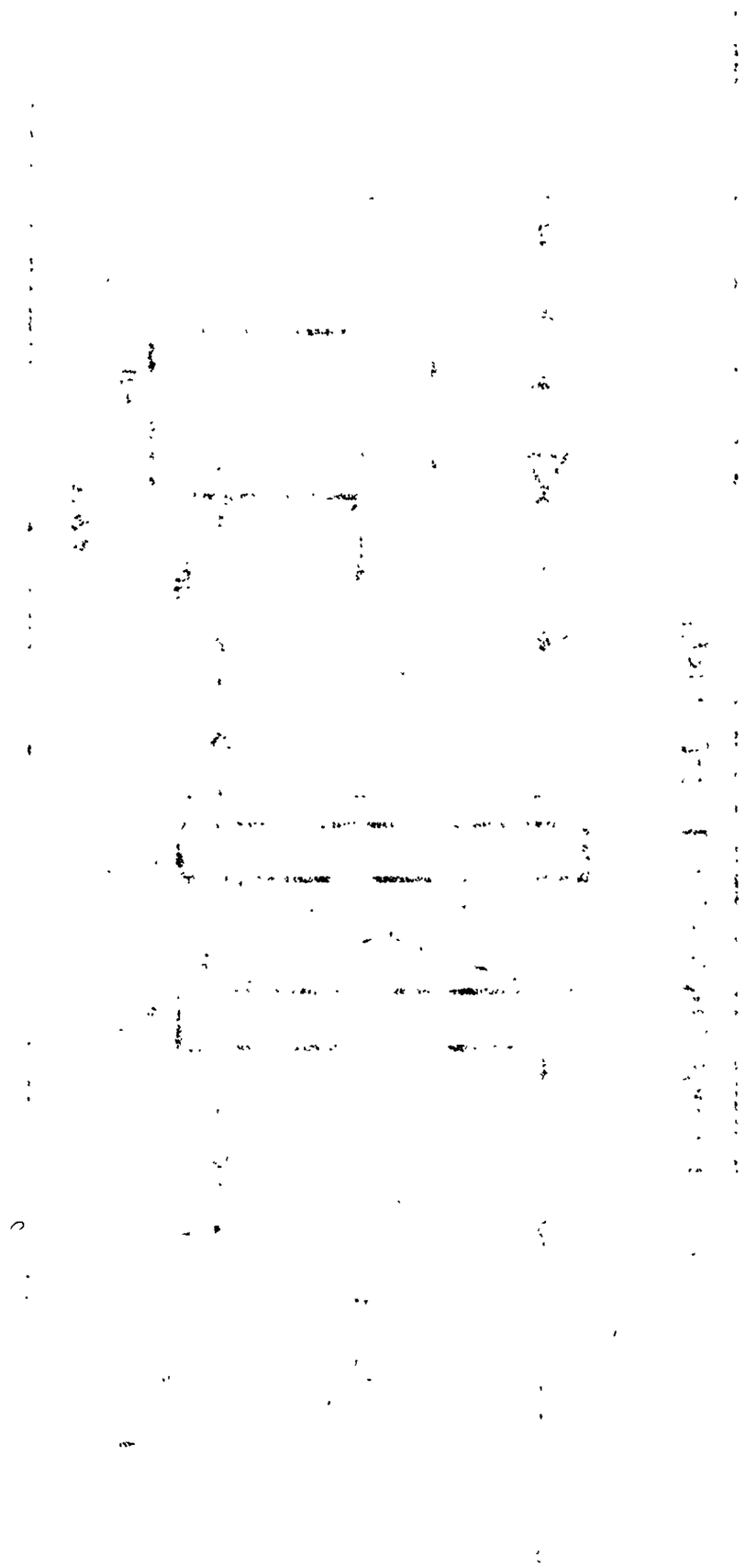


Figure 3-1. Data Set Simulator Cable

555-07



CHAPTER 4
IOC I/O INTERFACES

The IOC consists of the hardware and software that are used to establish interfaces between a master processor and both the integral CRT terminal and the integral diskette drive. Modification of the IOC is unnecessary since most IOC functions are integrated with Intel-supplied system software. This chapter provides details of IOC-master processor protocol as a basis for user-designed master programs.

4.1 IOC/MASTER PROCESSOR PROTOCOL

All communications between the IOC and the master processor are accomplished via the data bus buffer (DBB) of the IOC. The DBB is essentially an inter-bus communications facility that stores one input byte, one output byte, and one byte of DBB status. The DBB status bits indicate the presence of data in the input and output buffers, the busy status of the IOC processor relative to command processing, and the type of byte (command or data) present in the input/output buffer.

When the IOC is not busy processing a previously issued command, it is idle only in the sense that the IOC processor is not concerned with inter-processor communications. During this time, RAM refresh and CRT refresh cycles continually occur. Also, if any keyboard entries are made, the character bytes are saved by the keyboard processor. The master processor must periodically poll the IOC to determine if keyboard entries have occurred and then command the IOC to transfer the keyboard characters to the master via the IOC's DBB.

The commands used to input keyboard characters are functionally similar to commands used to update the CRT display or to accomplish diskette data transfers in that any command must be completely processed by the IOC before a new command can be issued. However, keyboard entries may occur at any time irrespective of I/O activities initiated by the master. Any program controlling the IOC must allow time for the acceptance of new keyboard entries during the execution of any I/O operation of long duration. The frequency of keyboard input commands must be sufficient to match the keystroke rate of a fast typist; the keyboard processor provides temporary storage for up to eight characters. Keyboard entries must be handled as unscheduled real-time events by the controlling resident or non-resident program.

4.2 DATA BUS BUFFER

The IPB/IPC (or any other master processor) uses two I/O ports for communications with the IOC. The first of these ports (port C0) is a data port that provides for single byte transfers to or from the IOC. The bytes may contain data to or from a device, status from the IOC or one of its devices, or diagnostic instructions or results from the IOC. The second port (port C1) is a control port that provides for command transfers to the IOC and the return of DBB status. The commands directly control the IOC and identify the type of data, status, or diagnostic information that is to be transferred via the data port. All transfers via the data port require prior master processor issuance of a command to the IOC control port. However, a command does not necessarily result in a data port transfer.

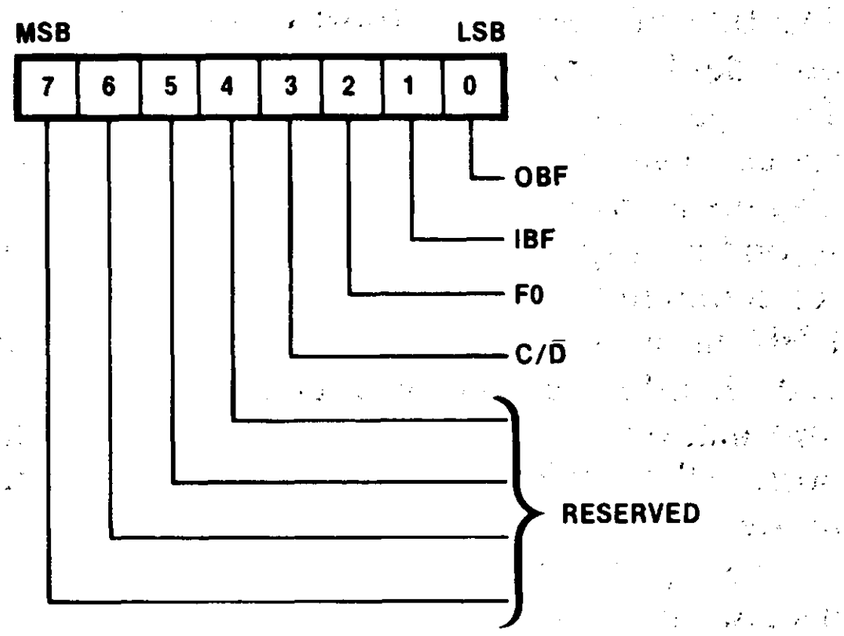
The preceding arrangement allows for the following types of transfers to and from the IOC:

- Direct transfer of DBB status without IOC participation.
- Direct control of the IOC by means of commands that do not result in a data transfer.
- Data transfers to or from a device on command.
- Status transfers from the IOC or a device on command.
- Diagnostic data transfers from the IOC on command.

Two important factors are associated with transfers between a master processor and the IOC. The first factor is that most commands cause the transfer of only one byte of data, status, or diagnostic information via the data port. If a block of information (other than diskette data) is to be transferred, a separate command is required for each byte of the block. The diskette data transfer commands (read and write) cause the transfer of a block of data to or from the master processor. The second factor is that the master processor maintains total control of transfers via the data port. The IOC, when responding to a read command, sets the F0 flag while it is executing the command and then sets the output buffer full (OBF) flag to indicate when the requested byte can be read by the master processor. Accordingly, the master processor must repeatedly access the DBB status byte to determine when the the input data is ready. Similarly, when the IOC is responding to a write command, it sets the F0 flag while it is executing the command and clears the input buffer full (IBF) flag when it accepts the byte (the master

processor sets the IBF flag when it writes the byte to the input buffer). The DBB status byte must also be accessed prior to issuing any command to ensure that the IOC is ready to accept the command (i.e., IOC busy flag F0 must be tested).

The format of the DBB status byte returned during an I/O read of port C1 is as follows:



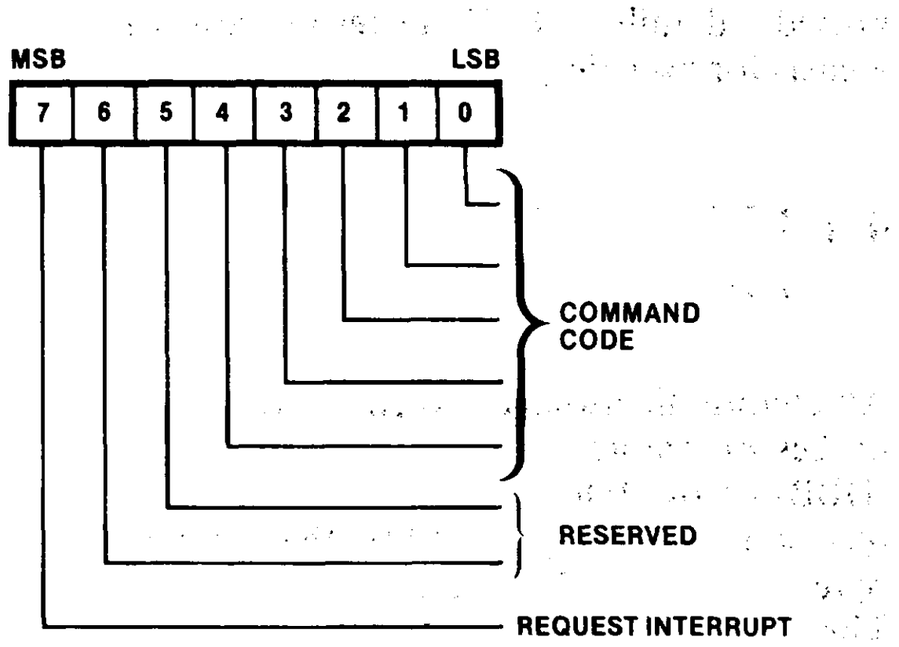
OBF Output Buffer Full. The OBF flag is automatically set (to a "1" state) by the IOC processor when the IOC writes a data byte to the output buffer. The OBF flag is automatically cleared (to a "0" state) when the master processor reads the byte from the output buffer.

IBF Input Buffer Full. The IBF flag is automatically set by the master processor when it writes a data byte to the input buffer. The IBF flag is automatically cleared when the IOC processor reads the byte from the input buffer.

F0 F0 flag. The F0 flag is set by the IOC processor on receipt of a command from the master processor in order to lockout additional command entry. On completion of the command, the IOC processor clears the F0 flag. The master processor monitors the F0 flag to determine when a command has been accepted (F0 flag set) and when command processing is complete (F0 flag clear).

C/$\bar{D}$ Command/Data. The C/$\bar{D}$ flag reflects the state of the master processor's low-order port address bit to differentiate between the writing of a data byte to port C0 (C/$\bar{D}$ = 0) and the writing of a command byte to port C1 (C/$\bar{D}$ = 1). The IOC processor examines this flag to determine if the byte in the input buffer is a command or data. The IOC processor also controls this flag to inform the master processor of the contents of the output buffer (if C/$\bar{D}$ = 0, the output buffer contains the requested data byte; if C/$\bar{D}$ = 1, the output buffer contains a status byte).

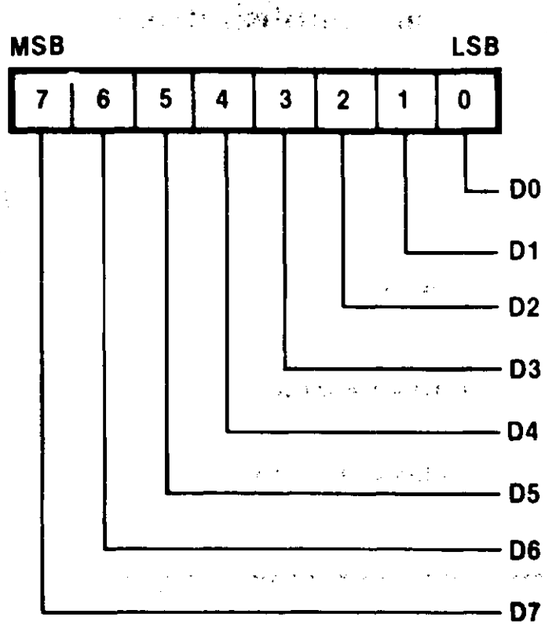
Command bytes transferred to the IOC during an I/O write to port C1 have the following general format:



Command Code is a 5-bit binary value that uniquely identifies each of the commands that may be issued by the master processor.

Request Interrupt is a control bit that informs the IOC that an interrupt is expected at the completion of the operation specified by the command. Commands that make use of the request interrupt bit include all commands that pass data to or from the CRT and the integral diskette.

Data, status and diagnostic bytes transferred via I/O port C0 have no specific format except as required by the associated command. Data bit mnemonics are as follows:



4.3 IOC COMMANDS

Specific commands are used with the diskette drive, the CRT, and the keyboard. The status byte returned by the individual I/O devices serves to verify proper operation of the device. Other commands are not used by a specific device and are known as system commands (all diagnostic commands are system commands). A complete listing of the IOC commands is provided in table 4-1.

4.3.1 SYSTEM COMMANDS

Eleven of the commands that may be issued by a master processor to the IOC are used to control or test subsystem functions that are common to all of the IOC devices. These system commands permit program-controlled hardware resetting, provide for the return of device and subsystem status, control enabling and resetting of interrupts, and enable diagnostic testing of IOC facilities. The following text describes each of the system commands and defines the format of data bytes that are transferred as a result of command execution.

NOTE

Command bit 7 has no function within the system commands (i.e., interrupts cannot be generated by the IOC on execution of a system command).

The PACIFY command is a software reset that terminates any pending I/O operation and reinitializes the IOC hardware and software. No data byte transfer is associated with this command. IOC initialization requires a minimum of 100 milliseconds, and no subsequent commands should be issued during this period.

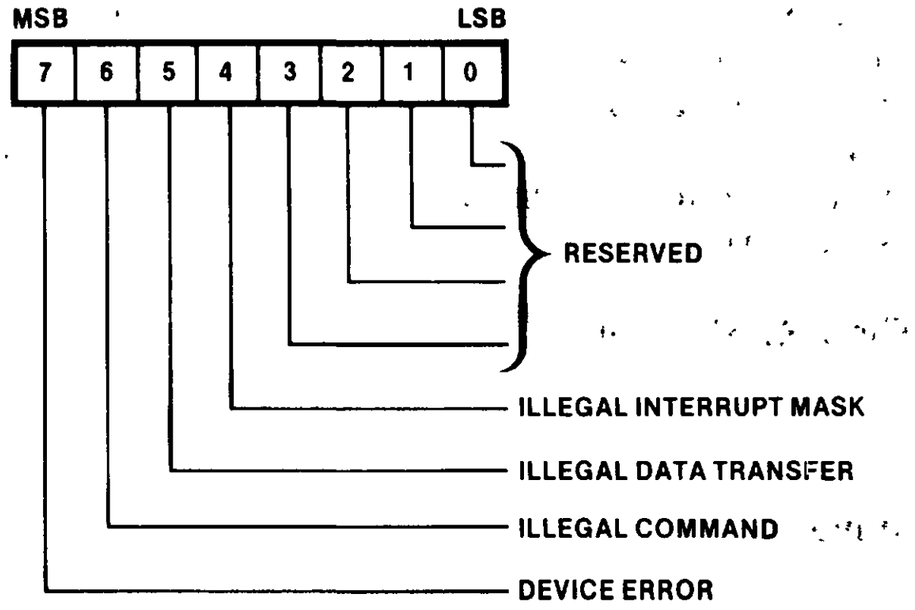
The ERESET command is intended for use with an I/O device that requires a hardware error reset to clear an error condition within the device. Since the

Table 4-1. IOC Command Set

Type	Command Code	Mnemonic	Function
System	00000	PACIFY	Resets IOC and its devices.
	00001	ERESET	Resets device-generated error (not used by standard devices).
	00010	SYSTAT	Returns subsystem status byte to master.
	00011	DSTAT	Returns device status byte to master.
	00100	SRQDAK	Enables input of device interrupt acknowledge mask from master.
	00101	SRQACK	Clears IOC subsystem interrupt request.
	00110	SRQ	Tests ability of IOC to forward an interrupt request to the master.
	00111	DECHO	Tests ability of IOC to echo data byte sent by master.
	01000	CSMEM	Requests IOC to checksum on-board ROM. Returns pass/fail.
	01001	TRAM	Requests IOC to test on-board RAM. Returns pass/fail.
	01010	SINT	Enables specified device interrupt from IOC.
	01011 thru 01111	—	Reserved, causes illegal command error.
CRT	10000	CRTC	Requests data byte output to the CRT monitor.
	10001	CRTS	Returns CRT status byte to master.
Keyboard	10010	KEYC	Requests data byte input from the keyboard.
	10011	KSTC	Returns keyboard status byte to master.
	10100	—	Reserved.
Integral Diskette	10101	WPBC	Enables input of first of five bytes that define current diskette operation.
	10110	WPBCC	Enables input of each of four bytes that follow WPBC.
	10111	WDBC	Enables input of diskette write bytes from master.
	11000	—	Reserved.
	11001	RDBC	Enables output of diskette read bytes to master.
	11010	—	Reserved.
	11011	RRSTS	Returns diskette result byte to master.
	11100	RDSTS	Returns diskette device status byte to master.
	11101 thru 11111	—	Reserved, causes illegal command error.

standard devices of the IOC do not require an error reset signal, the ERESET command is not implemented by IOC firmware.

The SYSTAT command causes the IOC processor to load the system status byte into the output data buffer of the DBB. The IOC processor sets the OBF flag and clears the C/$\overline{D}$ flag to inform the master processor that system status byte can be read from the data port. The format of the system status byte is as follows:



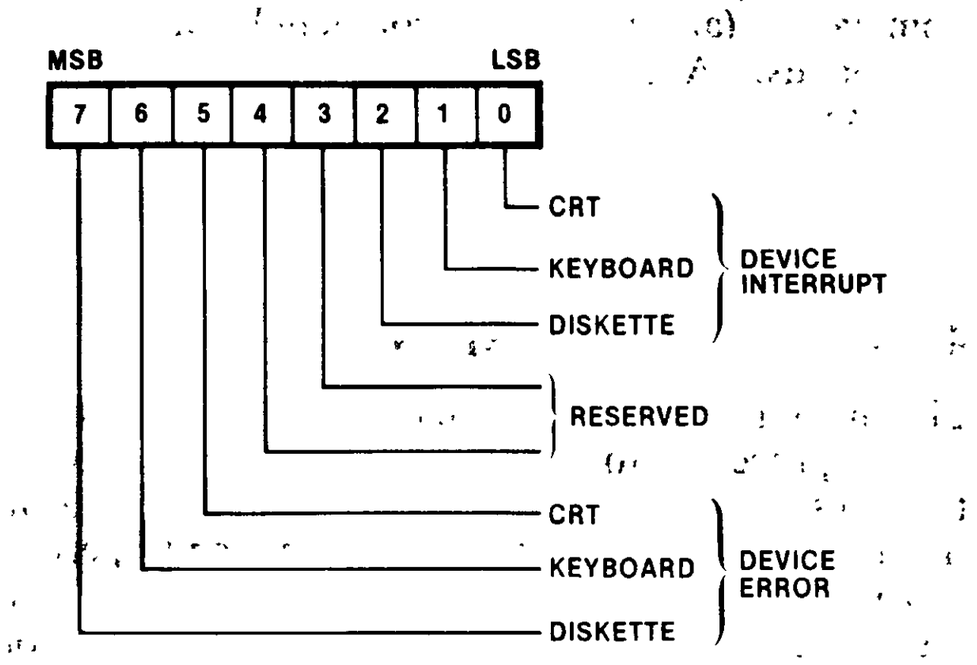
Illegal Interrupt Mask is set when the interrupt reset mask transferred by a SRQDAK command does not correspond to the interrupt bit set in the device status byte. The illegal interrupt mask bit is cleared when the master processor reads the system status byte from the output buffer.

Illegal Data Transfer is set when the master processor loads a data byte into the DBB input buffer without a preceding command. The data byte is not accepted by the IOC. The illegal data transfer bit is cleared when the master processor reads the system status byte from the output port.

Illegal Command is set when the master processor loads an undefined command code into the DBB input buffer (see table 4-1). The command is not executed by the IOC. The illegal command bit is cleared when the master processor reads the system status byte from the output buffer.

Device Error is set when a device fails to respond to a command. The master processor must issue a DSTAT command to determine the individual device responsible for the error. The device error bit is cleared by the DSTAT command.

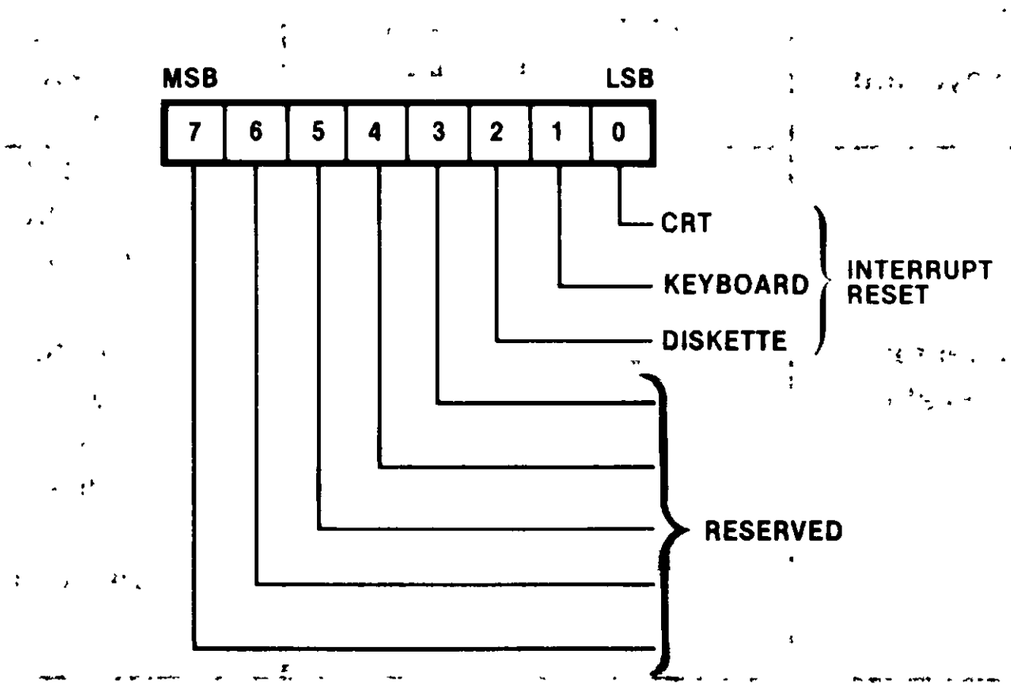
The DSTAT command causes the IOC processor to load the device status byte into the output data buffer of the DBB. The IOC processor sets the OBF flag and clears the C/$\overline{D}$ flag to inform the master processor that the device status byte can be read from the data port. The format of the device status byte is as follows:



A Device Interrupt bit is when the operation specified by a command has been completed and interrupts for the device have been enabled (request interrupt bit in the command byte set or device interrupt previously enabled by a SINT command). A device interrupt bit is cleared by a SRQDAK or SRQACK command.

A Device Error bit is set when the specified device fails to respond to a command issued by the master processor. A device error bit is cleared when the master processor reads the device status byte. More detailed error information is provided by the CRTS command (for CRT errors), the KSTC command (for keyboard errors) or the RDSTS command for diskette errors).

The SRQDAK command is used to clear a device interrupt. The subsequent data byte from the master processor to the input data buffer is as follows:



An Interrupt Reset bit, when set, clears the corresponding interrupt bit in the device status byte (see DSTAT command). Attempting to reset an interrupt bit that is not set causes the illegal interrupt mask bit to be set in the system status byte. Note that the SRQDAK command also clears the hardware interrupt to the IPB/IPC.

The SRQACK command causes the IOC to reset all of the interrupt bits in the device status byte and the hardware interrupt to the IPB/IPC. A data byte transfer is not associated with the SRQACK command.

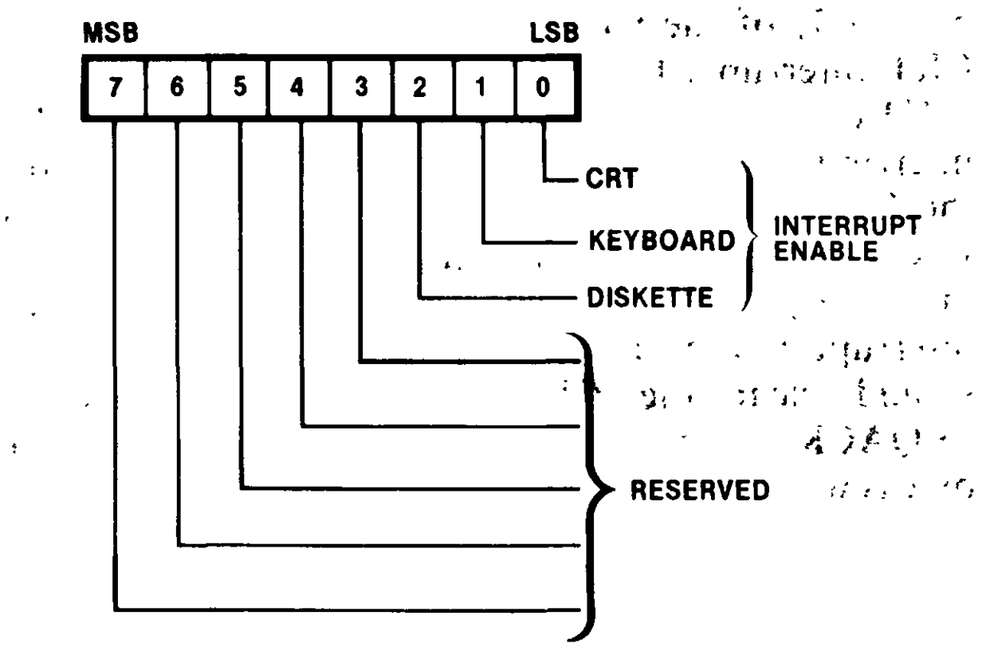
The SRQ command causes the IOC to generate a hardware interrupt to the IPB/IPC. The interrupt bits of the device status byte are not affected, and a data byte transfer is not initiated. This diagnostic command allows any master processor to test the IOC interrupt request line when all other local interrupts of the IPB/IPC are reset. The IOC interrupt request causes a level 7 interrupt request on the Multibus interface. The interrupt request is cleared by a SRQACK command.

The DECHO command causes the IOC processor to accept and return (in complemented form) the next data byte input from the master processor. Although the data byte is sent and received via I/O port C0, the path within the IOC includes a software-controlled transfer of the data byte from the DBB input data buffer to the DBB output data buffer. The entire action constitutes a fairly comprehensive test of the master processor-IOC interface. The IOC response to a DECHO command requires approximately two milliseconds.

The CSMEM command causes the IOC processor to checksum the contents of the IOC ROM and thereby perform a confidence test on the IOC firmware. If the checksum test passes, the IOC processor sets the C/$\overline{D}$ flag to zero and returns a data byte of all zeroes; if the checksum test fails, the IOC processor sets the C/$\overline{D}$ flag to one and returns a data byte of all ones. Command execution requires approximately 100 milliseconds.

The TRAM command causes the IOC processor to perform read-after-write testing of IOC RAM. If a RAM location is found to be faulty, the test is terminated, and the IOC processor sets the C/$\overline{D}$ flag to one and returns a data byte of all ones. If the test passes, the IOC processor sets the C/$\overline{D}$ flag to zero and returns a data byte of all zeros. Faulty locations are not identified. Command execution requires approximately 100 milliseconds.

The SINT command causes the IOC processor to accept an interrupt enable byte at the input data buffer of the DBB. The enabling of any interrupt bit also enables the hardware interrupt line (IOC interrupt) to the IPB/IPC. The interrupt enable bits perform a function identical to the request interrupt control bit (bit 7) of a command byte. Note that once an interrupt is enabled, it remains enabled until a subsequent SINT command is issued to clear the interrupt enable bit. The format of the interrupt enable byte is as follows:



An Interrupt Enable bit, when set, enables the interrupt from the corresponding IOC device. The format of the interrupt enable byte is identical to that of the interrupt reset byte of the SRQDAK command. Note that when any of the interrupts are enabled, the IOC interrupt to the IPB/IPC is also enabled.

4.3.2 CRT COMMANDS

All timing and formatting of the integral CRT display is established by the IOC firmware and the 8275 programmable CRT controller. The presentation of data is accomplished by transferring data from IOC RAM tables to the CRT character generating circuits. Commands from the master processor merely update the data tables. A single type of command is used to write keyboard inputs and to program responses on the CRT. A second CRT-associated command returns CRT status to the master processor.

The CRTC command causes the IOC to use the next data byte appearing at the DBB input data buffer as an input to the CRT display tables located in IOC RAM. The occurrence of the command is totally asynchronous with respect to CRT display raster timing. Furthermore, positioning of characters on the CRT screen is determined solely by a pointer that is maintained by the IOC firmware. Operator keyboard

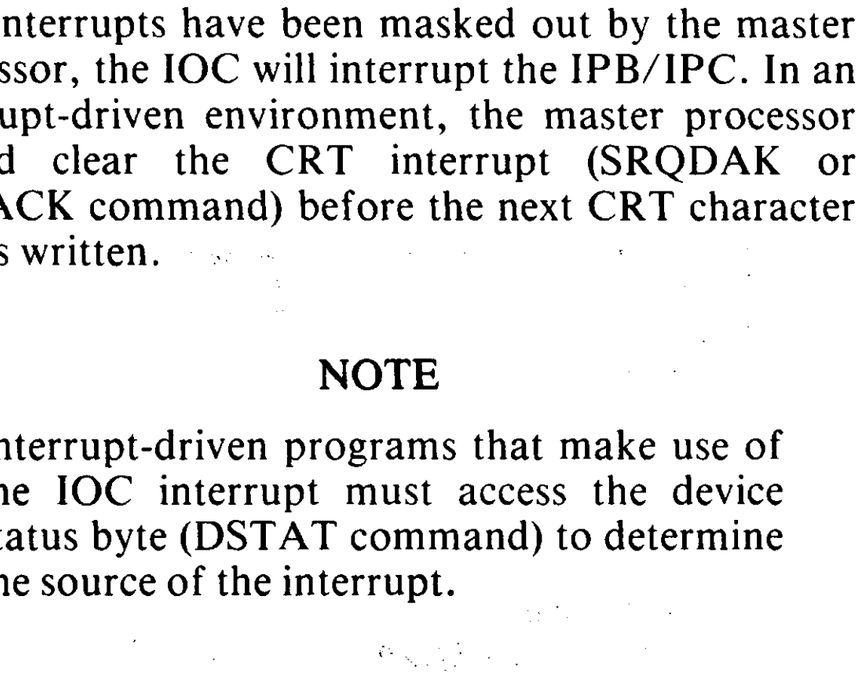
entries can alter the pointer for editing purposes, but the interpretation of the special characters used for this purpose (or any other purpose) is a function of the IOC firmware. The character codes employed are those of the ASCII set.

When operating under ISIS or Monitor, the IPB/IPC polls the DBB to determine when the IOC has accepted the CRT character byte, and interrupts are not employed (the request interrupt bit of the command byte is *not* set). When the request interrupt bit (bit 7) of the CRTC command byte is set (or if CRT interrupts have been previously enabled by a SINT command), the IOC processor will set the CRT interrupt bit in the device status byte when it writes the CRT character byte into IOC RAM and, unless IOC interrupts have been masked out by the master processor, the IOC will interrupt the IPB/IPC. In an interrupt-driven environment, the master processor should clear the CRT interrupt (SRQDAK or SRQACK command) before the next CRT character byte is written.

NOTE

Interrupt-driven programs that make use of the IOC interrupt must access the device status byte (DSTAT command) to determine the source of the interrupt.

The CRTS command causes the IOC processor to load the CRT status byte into the output data buffer of the DBB and to clear the CRT device error bit in the device status byte. The CRTS command is normally issued only in response to a CRT error as indicated by the setting of the device error bit in the system status byte and the setting of the CRT error bit in the device status byte. The format of the CRT status byte is as follows:



CRT Present is set during initialization and indicates that the IOC is operational (i.e., able to respond to commands from a master processor).

Illegal Data is set when the master processor loads a data byte into the DBB input buffer without a preceding CRTC command. The illegal data bit is cleared when the master processor reads the CRT status byte.

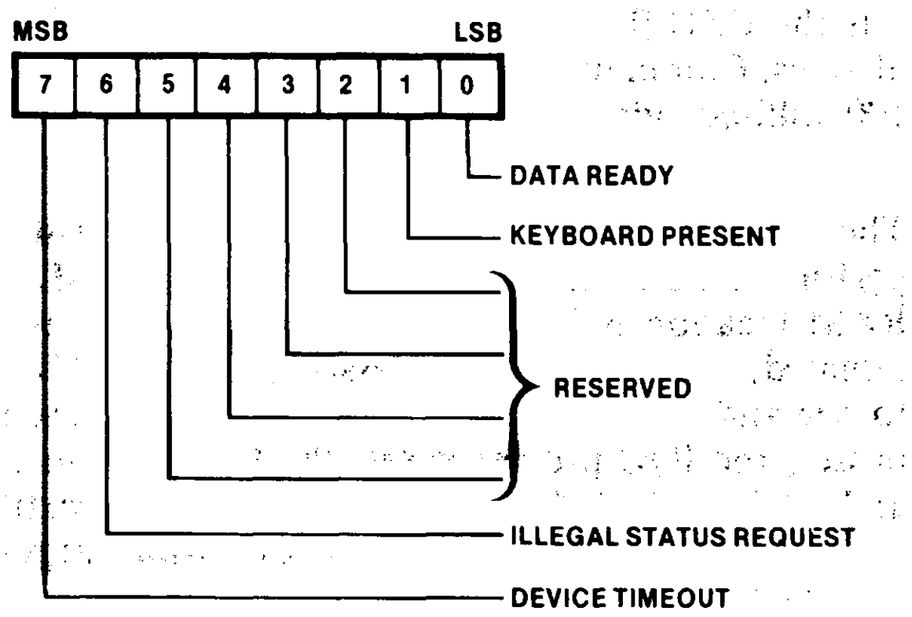
Illegal Status is set when the request interrupt bit (bit 7) of the CRTS command byte is erroneously set by the master processor. (A master processor cannot legally request an interrupt at the completion of a status accessing operation.) The illegal status bit is cleared when the master processor reads the CRT status byte.

4.3.3 KEYBOARD COMMANDS

Two commands are used with the keyboard; one command is used to access keyboard entries, and the other command is used to access keyboard status.

The KEYC command causes the IOC processor to access an ASCII character byte from the keyboard processor and to place this byte in the output data buffer. The KEYC command is the only IOC data transfer command that does not make use of the request interrupt bit in the command byte (to enable keyboard interrupts, the SINT command must be used). The format for the keyboard character bytes is established by the ASCII standard.

The KSTC command causes the IOC to access the keyboard status byte and to place this byte in the output data buffer of the DBB. When operating in the polled mode (interrupts disabled), the master processor first uses the KSTC command to determine when a character has been entered at the keyboard (by testing the data ready bit) and then uses the KEYC command to access the character byte. The KSTC command is also issued in response to a keyboard error (indicated by the setting of the device error bit in the system status byte and the setting of the keyboard device error bit in the device status byte). The format of the keyboard status byte is as follows:



Data Ready is set when a character byte is available from the keyboard processor as a result of keyboard entry.

Keyboard Present is set when the keyboard is connected to the development system. The IPB/IPC examines this bit during initialization to assign the system console device (if the keyboard present bit is set, the keyboard and integral CRT are assigned as the system console; if the keyboard present bit is clear, the device attached to one of the serial I/O channels is assigned as the system console).

Illegal Status Request is set when the request interrupt bit in the KSTC command byte is set (a master processor cannot legally request an interrupt at the completion of a status-access operation) and is cleared when the master processor reads the keyboard status byte.

Device Timeout is set when a KEYC command is issued when a character byte is not available from the keyboard processor (data ready bit clear). The device timeout bit is cleared when the master processor reads the keyboard status byte.

4.3.4 INTEGRAL DISKETTE COMMANDS

The integral diskette of the Intellec Series II development system is a random-access, mass-storage media on which information is formatted to simplify access and to utilize available storage space. Accordingly, a significant amount of control information must be passed to the diskette and its control circuits. Some control information such as sector size may be fixed and can be established during system initialization. Other information, such as the size of a file to be recorded and the availability of storage space, can only be determined immediately prior to recording the file and requires active participation on the part of the master program.

The need to provide for master program control of diskette recording plus the fact that data transfers occur in two directions (diskette read or write) substantially increases the number of commands that must be issued to the IOC. Furthermore, two types of status bytes are returned by the IOC processor; one to indicate the success of IOC/master interactions and the other to indicate the results of IOC/diskette drive interactions. The six commands required to transfer diskette control, data, and status bytes are listed in table 4-1.

Most of the control information supplied to the IOC is provided in the form of five bytes that are referred to as the I/O parameter block (IOPB). These bytes specify the diskette operation to be performed and provide formatting information including the number of records (sectors) to be transferred and the track and sector addresses.

The diskette commands are associated with two interfaces; the interface between the IOC and a master processor and the interface between the IOC processor and the diskette. The diskette read and write commands (RDBC and WDBC) transfer data between the master processor and IOC RAM. Conversely, the parameter block write commands (WPBC and WPBCC) load the I/O parameter block into IOC RAM; the contents of the parameter block are used to define and initiate data transfers between IOC RAM and the diskette. The read drive status command (RDSTS), in addition to defining the current status of the drive, indicates incorrect command entry or execution. The read result status command (RRSTS) is used to verify the result of a diskette operation.

Diskette data transfers must be viewed as two separate operations; the transfer of the data block between a master processor and IOC RAM and the transfer of the data block between IOC RAM and the diskette. Data block transfers between IOC RAM and the diskette are automatically initiated when the last byte of the I/O parameter block is written into IOC RAM. Accordingly, in order to write data on the diskette, the data block must first be written into IOC RAM before the parameter block is written. Conversely, in order to read a data block from the diskette, the parameter block must precede the reading of the data block from IOC RAM. Typical command sequences for diskette read and write operations are outlined in table 4-2.

The WPBC command causes the IOC processor to accept the next data byte at the DBB input data buffer as the first (channel word) byte of a parameter block. The IOC processor writes this data byte into a preassigned location in IOC RAM. The request interrupt bit of the command byte, if set, causes a diskette device interrupt to be generated when the byte is written into RAM. The format and function of the first parameter block byte are discussed following the description of the WPBCC command.

The WPBCC command causes the IOC processor to accept the next data byte at the DBB input data buffer as one of the subsequent I/O parameter block bytes. The IOC processor writes the byte into IOC RAM and, if the request interrupt bit is set in the

Table 4-2. Typical Diskette Read and Write Command Sequences

Read Sequence	
Command	**Action**
RDSTS	Determine if drive is ready
WPBC	Input IOPB Channel Word byte
WPBCC	Input IOPB Diskette Instruction byte
WPBCC	Input IOPB Sector Count byte
WPBCC	Input IOPB Track Address byte
WPBCC	Input IOPB Sector Address byte
RDSTS	Determine when operation is complete (polled mode)
RRSTS	Determine if operation was successful
RDBC	Output data block from IOC RAM

Write Sequence	
Command	**Action**
RDSTS	Determine if drive is ready
WDBC	Input data block to IOC RAM
WPBC	Input IOPB Channel Word byte
WPBCC	Input IOPB Diskette Instruction byte
WPBCC	Input IOPB Sector Count byte
WPBCC	Input IOPB Track Address byte
WPBCC	Input IOPB Sector Address byte
RDSTS	Determine when operation is complete (polled mode)
RRSTS	Determine if operation was successful

NOTE:

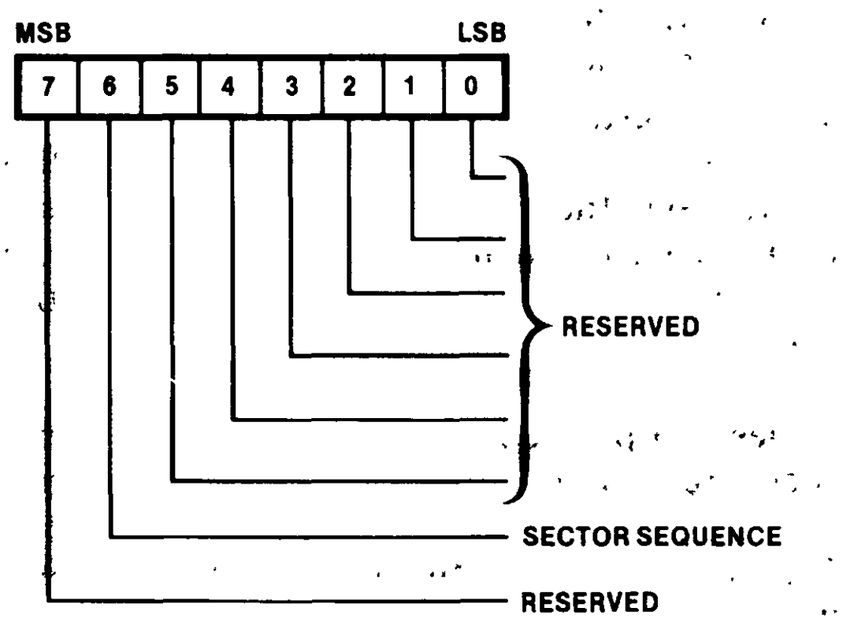
1. First byte of data block input to IOC from master processor contains sector count that is multiplied by 128 by the IOC to determine the number of bytes to be transferred.

command byte, generates a diskette device interrupt when the byte is written. The data byte being written into IOC RAM is either the second, third, fourth or fifth byte of the I/O parameter block. The byte written and the IOC RAM location selected are determined by the occurrence of the WPBC command and any subsequent WPBCC command. In other words, the WPBC command serves as a reference for the I/O parameter block and the following four WPBCC commands load the four remaining parameter bytes in consecutive RAM locations. Accordingly, the parameter block bytes must be presented in sequence, and any detected error requires reentry of the entire I/O parameter block. The sequence in which I/O parameter block bytes are input is indicated in table 4-2.

NOTE

Further details on programming diskette operations are provided in the 8271 Programmable Floppy Disk Controller data sheet.

The format of the I/O parameter block Channel Word byte (byte 1) associated with the WPBC command is as follows:

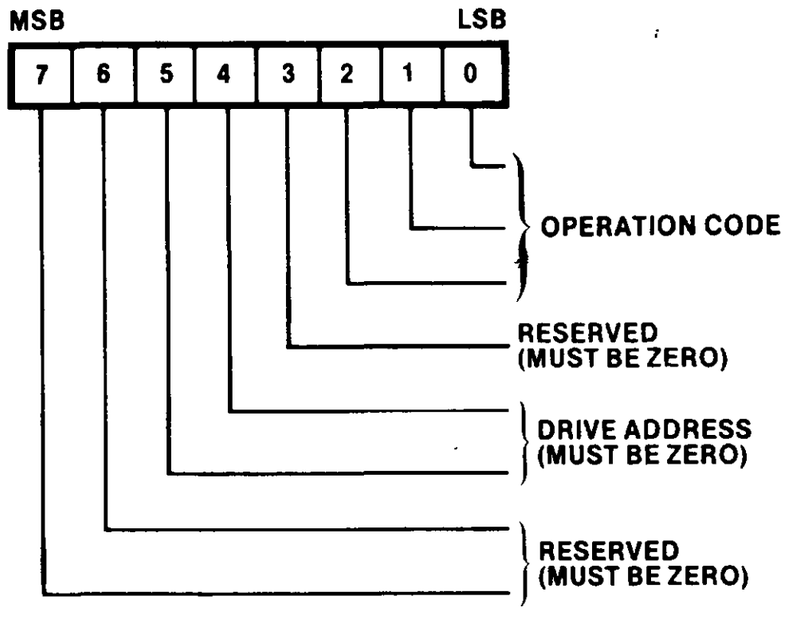


The sector sequence bit (bit 6) is only examined when a format track operation is specified in the diskette instruction byte (byte 2) of the I/O parameter block. When the sector sequence bit is set, the "random" sector format is selected, and the diskette controller

accesses a format table in IOC RAM to determine the order in which logical sector addresses are assigned when formatting the track. The format table consists of 26 data byte pairs. The first byte of each pair is a sector address ranging from 1 to 26 (01H-1AH); the second byte can contain any value and is required only to maintain compatibility with the Intel two-board diskette controllers. The format table must be written into IOC RAM (using the WDBC command) prior to initiating the format track operation. The format table itself begins at the same predefined location in IOC RAM that is used to temporarily store diskette data. The first entry in the table is the sector count byte (see WDBC command description) and contains a value of 01H (the IOC processor multiplies the sector count value by 128 to determine the number of consecutive RAM locations to be written by the WDBC command). During the format track operation, the diskette controller writes the first sector address entry from the table into the ID field of physical sector 01 (the first sector following the index mark), the second sector address table entry into physical sector 02, etc., until all 26 sectors have been written. The track address written into the ID field is taken directly from the track address byte (byte 4) of the I/O parameter block. The sector data marks, CRC characters and gaps are supplied by the diskette controller, and the data field of each sector is filled with 128 bytes of 0E5H.

When the sector sequence bit is clear, the "sequential" sector format is selected. The format track operation follows the random sector format description of the previous paragraph with the exception that the sector addresses are assigned in sequential order (i.e., physical sector 01 is assigned logical sector address 01, physical sector 02 is assigned logical sector 02, etc.) by the diskette controller, and a format table is not required.

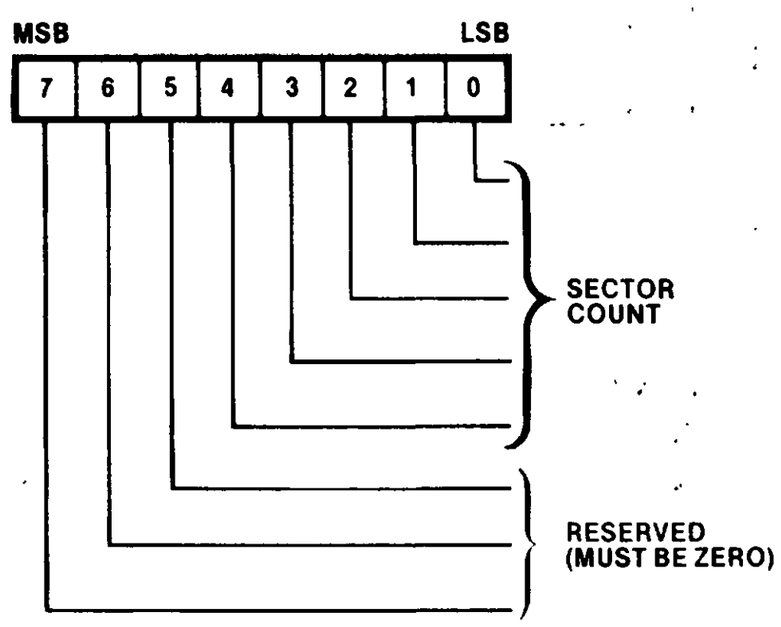
The format of the I/O parameter block Diskette Instruction byte (byte 2) is as follows:



Drive Address. Bits 4 and 5 specify the drive address. The integral drive is assigned unit 0, and bits 4 and 5 must both be zero.

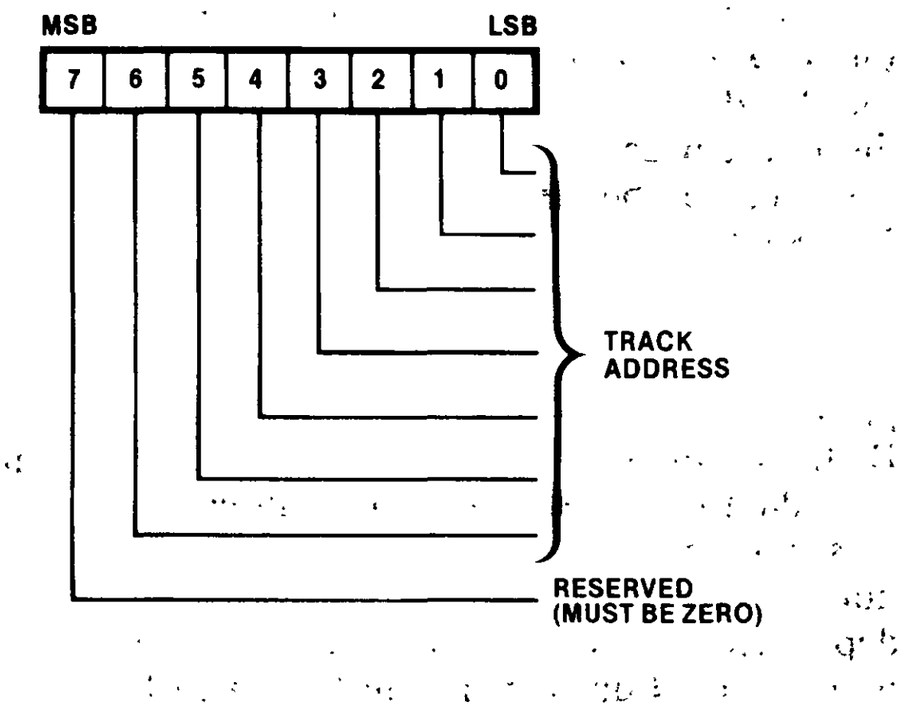
Operation Code. Bits 0, 1 and 2 specify the diskette operation to be performed. Operation code functions are outlined in table 4-3.

The format of the I/O parameter block Sector Count byte (byte 3) is as follows:



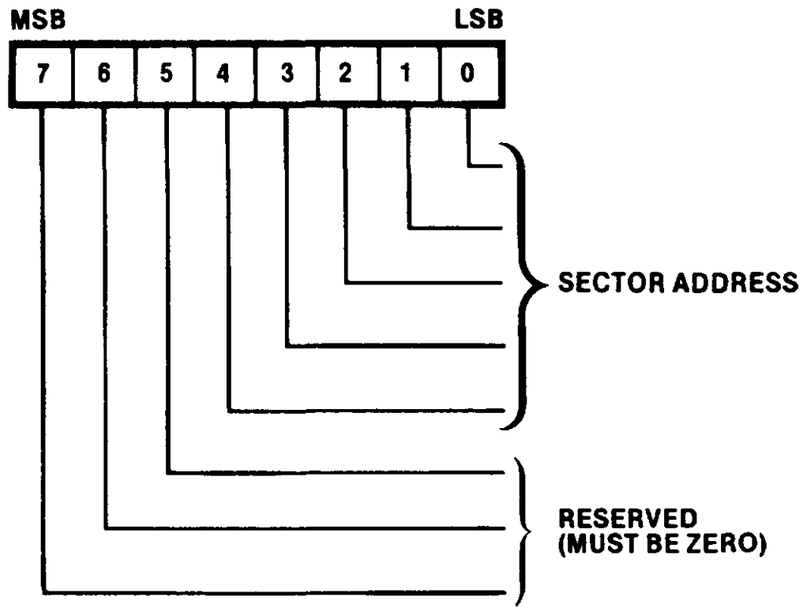
The Sector Count byte specifies the number of sectors to be accessed during a diskette read, write or verify operation. The combined value of the sector count and the (starting) sector address (byte 5 of the I/O parameter block) cannot be greater than 26 (1AH). Note that if the sector count value is 0H, one sector is transferred.

The format of the I/O parameter block Track Address byte (byte 4) is as follows:



The Track Address byte specifies the track to be accessed during a subsequent diskette seek, read or write operation. Since a diskette has 77 tracks, legal values range from 00H to 4CH (tracks 0 through 76).

The format of the I/O parameter block Sector Address byte (byte 5) is as follows:

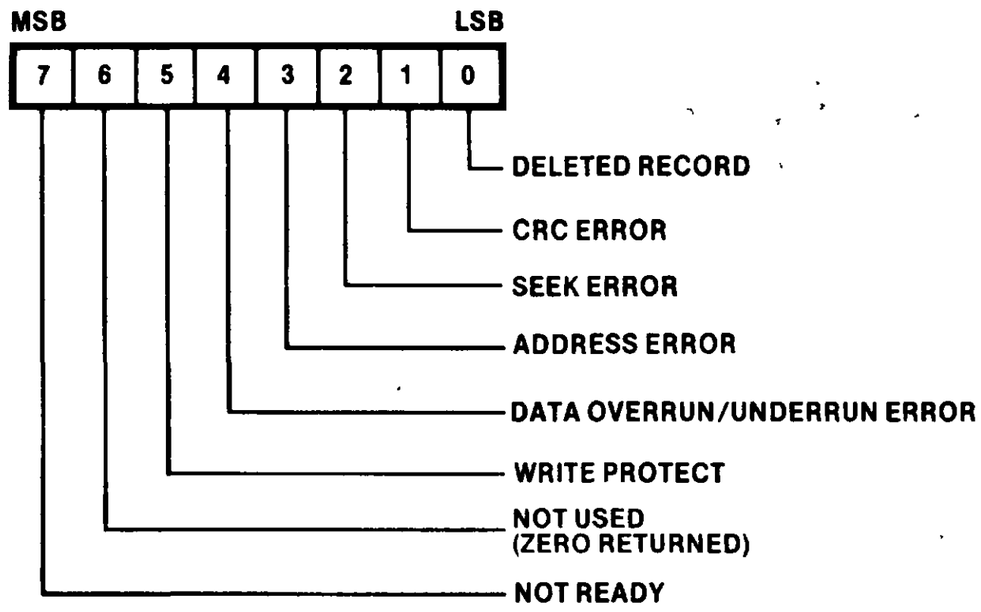


The Sector Address byte specifies the sector or the first of a series of sectors to be accessed during a subsequent diskette read, write or verify operation. Legal values range from 01H to 1AH (1-26).

The WDBC command causes the IOC processor to read-in a block of data bytes through the DBB input data buffer and to write the data bytes into sequential locations in IOC RAM. The WDBC command is used prior to a write data or format track (random format only) diskette operation to load the data to be written on the diskette into IOC RAM. The first byte transferred by the WDBC command is a count byte that specifies the number of 128-byte blocks to follow (the count byte is not written into RAM; the IOC multiplies the value by 128 to determine the extent of the transfer). Note that while the format table associated with the format track operation only requires 52 byte entries, 128 bytes are transferred by the WDBC command. The request interrupt bit of the WDBC command byte, when set, causes a diskette device interrupt to be generated when the specified number of data bytes have been written into IOC RAM.

The RDBC command causes the IOC processor to place sequential data bytes from IOC RAM into the DBB output data buffer and is issued following a read data diskette operation to allow a master processor to access the data read from the diskette. The number of bytes transferred by a RDBC command is dependent on the number of sectors read during the previous read data operation (specified by byte 3 of the associated I/O parameter block). When the request interrupt bit is set in the RDBC command byte, a diskette device interrupt is generated when the master processor reads-in the last data byte from the DBB output data buffer.

The RRSTS command is issued when a diskette device error is indicated in the device status byte and causes the IOC processor to place the diskette controller's result byte in the DBB output data buffer. The diskette device error bit is cleared when the master processor reads the result byte. The individual bits of the result byte are defined as follows:



Deleted Record is set when an attempt is made to read or verify a deleted sector (a sector that has previously been rewritten with a deleted data mark at the beginning of its data field). If the deleted record bit is set, the data is not transferred or verified.

CRC Error is set when the CRC character computed for a read data or verify CRC operation does not match the CRC character generated when the sector was written. Since this error is not detected until the sector is read, the data in IOC RAM must be considered invalid.

Seek Error is set during a read, write or verify operation when the addressed sector cannot be located within one complete revolution or when the track address specified does not match the track address read. If the seek error bit is set, no data is transferred, and a recalibrate operation should be performed to position the drive's read/write head at a known location.

Address Error is set when:

- the track address specified is greater than 76 (4CH).
- a sector address of 00H is specified.
- a sector address greater than 26 (1AH) is specified.
- the sector address and the number of sectors specified is greater than 26 (1AH).

Table 4-3. Diskette Operation Codes

Operation Code Bit 2	Bit 1	Bit 0	Operation	Function
0	0	0	No Operation	No operation. The diskette controller immediately sets the operation complete status bit in the diskette device status byte when the last I/O parameter block is written.
0	0	1	Seek	Initiates a seek operation to the track address specified in byte 4 of the I/O parameter block and sets the operation complete status bit in the diskette device status byte.
0	1	0	Format Track	Initiates a seek operation to the track address specified in byte 4 of the I/O parameter block. When the complete track has been formatted, the diskette controller sets the operation complete status bit.
0	1	1	Recalibrate	Steps the diskette drive's read/write head out until a track 0 indication is received from the drive. When track 0 is located, the diskette controller sets the operation complete status bit.
1	0	0	Read Data	Initiates a seek operation to the track address specified in byte 4 of the I/O parameter block and begins reading the sector ID fields until the sector addressed in byte 5 of the I/O parameter block is located. When the addressed sector is located, transfer the data contents of the number of sectors specified by byte 3 of the I/O parameter block to IOC RAM. When the transfer is complete, the diskette controller sets the operation complete status bit.
1	0	1	Verify CRC	Identical to the read data operation except that no data is transferred to IOC RAM. If an error is detected (i.e., if the data read does not match the data previously written), the diskette controller sets the CRC error bit in the result status byte. The operation complete status bit is set when the operation is complete (or if an error is detected).
1	1	0	Write Data	Initiates a seek operation to the track address specified in byte 4 of the I/O parameter block and begins reading the sector ID fields until the sector addressed in byte 5 of the I/O parameter block is located. When the addressed sector is located, reads in the data from IOC RAM and serially writes the data into the sector's data field. When the number of sectors specified by byte 3 of the I/O parameter block have been written, the diskette controller sets the operation complete status bit.
1	1	1	Write Deleted Data	Identical to the write data operation except that a deleted data mark is written in place of the data address mark at the beginning of the data field.

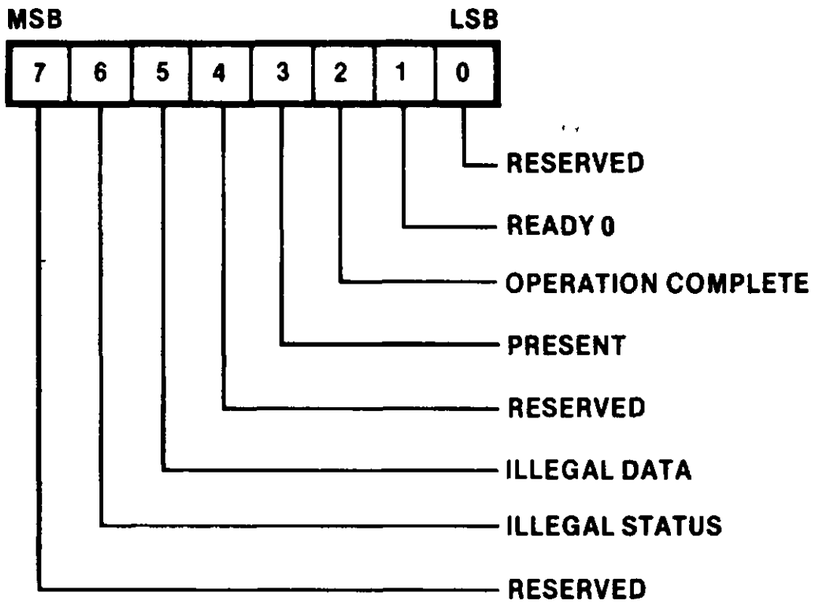
Data Overrun/Underrun Error is set when the diskette controller is unable to write a data byte to RAM before it is overwritten or when the requested data byte is not received from RAM in time to be written on the diskette.

Write Protect is set when an attempt is made to format or write to a write-protected diskette. When this bit is set, the operation is prevented, and no data is written on the diskette.

Not Ready is set when the diskette drive is not ready to perform a seek, read or write operation (i.e., the door is open or a diskette is not installed).

The RDSTS command causes the IOC processor to place the diskette status byte in the DBB output data buffer. The RDSTS command is used prior to a diskette operation to determine if the drive is ready

and is used in the polled mode (interrupts disabled) to determine when the operation has been completed. The format of the diskette status byte is as follows:



Ready 0 is set when the diskette is ready to perform a seek, read or write operation (i.e., the diskette is in place and up to speed).

Operation Complete is set when the specified operation has been completed or when the operation cannot be completed as a result of an error condition. When this bit is set, the device status byte and/or the controller's result byte should be examined.

Present is set when the integral diskette drive is physically connected to the IOC circuit board.

Illegal Data is set when the master processor loads a data byte into the DBB input data buffer without a preceding integral diskette command.

Illegal Status is set when the request interrupt bit (bit 7) of an RDSTS or RRSTS command is set by a master processor (a master processor cannot request an interrupt at the completion of a status-access operation).

CHAPTER 5
PIO SUBSYSTEM INTERFACES

The Parallel Input Output (PIO) subsystem consists of the hardware and software that are used to establish interfaces between a master processor and the standard parallel-byte peripheral devices of the Intellec Series II development systems. The standard peripherals consist of a line printer, a paper tape punch, a paper tape reader, and an Intel PROM programmer.

The interfaces for the printer and paper tape reader/punch are somewhat specialized, but may be modified to meet the requirements of similar devices. The bidirectional PROM programmer interface is more adaptable and can be modified to meet the parallel communication needs of a wide variety of intellegent devices, subsystems, and systems.

The PIO subsystem is, in several respects, similar to the IOC. The PIO subsystem includes a data bus buffer (DBB) that is functionally identical to the IOC's DBB; the PIO system commands are identical in function to the IOC system commands, and the PIO device commands are comparable to the IOC commands that control the CRT and keyboard. The device interfaces of the two subsystems differ, however, due to the specific requirements of the devices. All PIO functions are implemented within an Intel 8041 PIO processor. The 8041 chip contains the DBB as well as all data and program memory required for peripheral device control. Conversely, the PIO subsystem does not use intelligent hardware in the form of special- purpose programmable chips; all device signal timing and control are established by the PIO. The PIO is more flexible in terms of the special I/O device interfaces that may be implemented.

All communications between the PIO and a master processor are accomplished via the DBB. The DBB is essentially an interbus communications facility that stores one input byte or output byte and one byte of DBB status. The DBB status bits indicate the presence of data in the input and output buffers, the busy status of the PIO processor relative to command execution, and the type of byte (command/ status or data) in the input or output buffer.

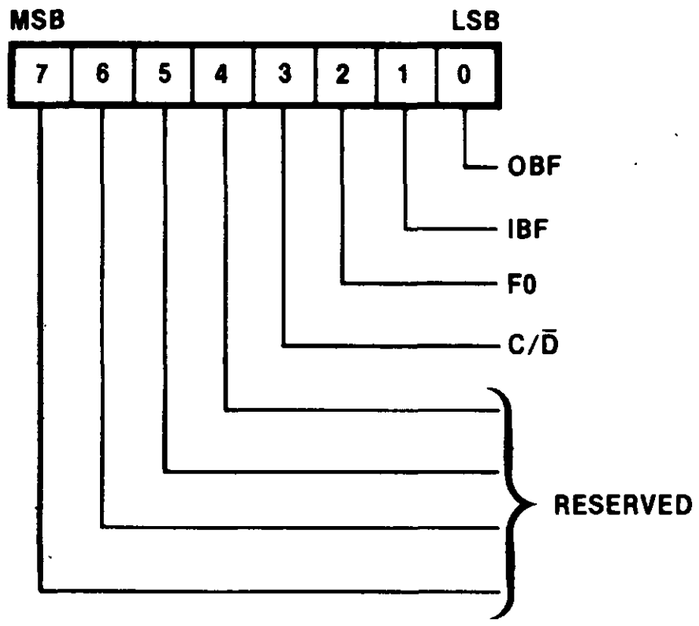
5.1 PIO/MASTER PROCESSOR PROTOCOL

The IPB/IPC (or any other master processor) uses two I/O ports for communications with the PIO processor's data bus buffer. One port (port F8) is a bidirectional data port that provides single byte transfers to or from the PIO processor's data bus buffer. The byte transferred may contain data to or from a device, status from the PIO processor or one of its devices, or diagnostic information from the PIO processor. The other port (port F9) is a control port that is used to transfer commands to the PIO processor or to return DBB status to the master processor. All transfers via the data port must be preceded by the issuance of a command to the PIO's control port, although a command does not necessarily result in a data port transfer.

Two facts should be noted regarding transfers between a master processor and the PIO processor. The first fact is that most commands transfer only one byte of data, status or diagnostic information via the data port, and in order to transfer a block of information, a separate command must be issued for each byte of the block (the PROM programmer read and write commands transfer three bytes). The second fact is that the master processor maintains complete control of all transfers via the data bus buffers; the PIO processor can only write to the DBB output data buffer when requested by a master processor.

The PIO processor, when responding to a command specifying a read operation, sets the F0 flag while it is executing the command and then sets the output buffer full (OBF) flag to indicate when the requested byte can be read by the master processor. Unless interrupts are employed, the master processor must repeatedly access (poll) the DBB status byte to determine when the requested input data is available. Similarly, when the PIO processor is responding to a command specifying a write operation, it sets the F0 flag while it is executing the command and clears the input buffer full (IBF) flag when it accepts the byte (the master processor sets the IBF flag when it writes the byte to the input buffer). The DBB status byte must also be examined prior to issuing any command to ensure that the PIO processor is ready to accept the command (i.e., PIO busy flag F0 must be tested).

The format of the DBB status byte returned during an I/O read of port F9 is as follows.



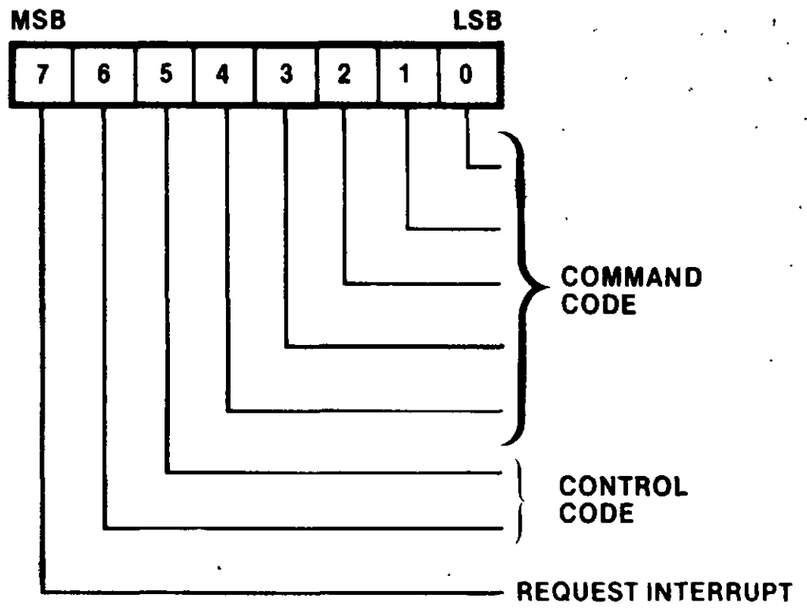
OBF Output Buffer Full. The OBF flag is automatically set (to a "1" state) by the PIO processor when it writes a data byte to the output buffer. The OBF flag is automatically cleared (to a "0" state) when the master processor reads the byte from the output buffer.

IBF Input Buffer Full. The IBF flag is automatically set by the master processor when it writes a data byte to the input buffer. The IBF flag is automatically cleared when the PIO processor reads the byte from the input buffer.

F0 F0 flag. The F0 flag is set by the PIO processor on receipt of a command from the master processor in order to lock out additional command entry. On completion of the command, the PIO processor clears the F0 flag. The master processor monitors the F0 flag to determine when a command has been accepted (F0 flag set) and when command processing is complete (F0 flag clear).

C/$\bar{D}$ Command/Data. The C/$\bar{D}$ flag reflects the state of the master processor's low-order port address bit to differentiate between the writing of a data byte to port F8 (C/$\bar{D}$=0) and the writing of a command byte to port F9 (C/$\bar{D}$=1). The PIO processor examines this flag to determine if the byte in the input buffer is a command or data. The PIO processor also controls this flag to inform the master processor of the contents of the output buffer (if C/$\bar{D}$=0, the output buffer contains the requested data byte; if C/$\bar{D}$=1, the output buffer contains a status byte).

Command bytes transferred to the PIO processor during an I/O write to port F9 have the following general format:



Command Code is a 5-bit binary value that uniquely identifies each of the commands that may be issued by the master processor.

Control Code is a two-bit code that provides additional control of PIO paper tape reader operations.

Request Interrupt is a control bit that informs the PIO that an interrupt is expected at the completion of the operation specified by the command. Commands that use the request interrupt bit are the paper tape reader, paper tape punch and line printer control commands.

5.2 PIO COMMANDS

Specific commands are used to transfer data between the master processor and the peripheral devices associated with the PIO subsystem and to return status bytes from the individual devices. Other commands are not used by any specific device and are defined as system commands. A complete list of the PIO commands is provided in table 5-1.

5.2.1 SYSTEM COMMANDS

There are eleven commands available to a master processor that are used to control or test system functions common to the PIO subsystem. These system commands are identical in function to the eleven system commands of the IOC and permit a program-controlled reset function, return device and subsystem status, control the enabling and disabling of interrupts and permit diagnostic testing of the PIO

Table 5-1. PIO Command Set

Command Type	Command Code	Mnemonic	Function
System	00000	PACIFY	Resets PIO and its devices.
	00001	ERESET	Resets device-generated error (not used by standard devices).
	00010	SYSTAT	Returns subsystem status byte to master processor.
	00011	DSTAT	Returns device status byte to master processor.
	00100	SRQDAK	Inputs device interrupt acknowledge mask from master processor.
	00101	SRQACK	Clears PIO subsystem interrupt request.
	00110	SRQ	Requests PIO to forward an interrupt request to the master processor.
	00111	DECHO	Requests PIO to echo data byte sent by master processor.
	01000	CSMEM	Requests PIO to checksum internal ROM. Returns pass/fail.
	01001	TRAM	Requests PIO to test internal RAM. Returns pass/fail.
	01010	SINT	Enables specified device interrupt from PIO.
	01011 thru 01111	—	Reserved, causes illegal command error.
Reader	10000	RDRC	Moves tape one frame forward/reverse or returns paper tape reader data byte.
	10001	RSTC	Returns paper tape reader status byte.
Punch	10010	PUNC	Transfers data byte to paper tape punch.
	10011	PSTC	Returns paper tape punch status byte.
Printer	10100	LPTC	Transfers data byte to printer.
	10101	LSTC	Returns printer status byte.
	10110	WPPC	Transfers two address/control bytes and one write data byte to PROM programmer.
PROM Programmer	10111	RPPC	Transfers two address/control bytes to PROM programmer and one read data byte from PROM programmer.
	11000	RPSTC	Transfers two PROM programmer status bytes to master.
	11001	RDPDC	Transfers read data byte from PROM programmer.
	11010 thru 11111	—	Reserved, causes illegal command error.

facilities. The following text describes each of the system commands and defines the format of the data bytes transferred as a result of command execution.

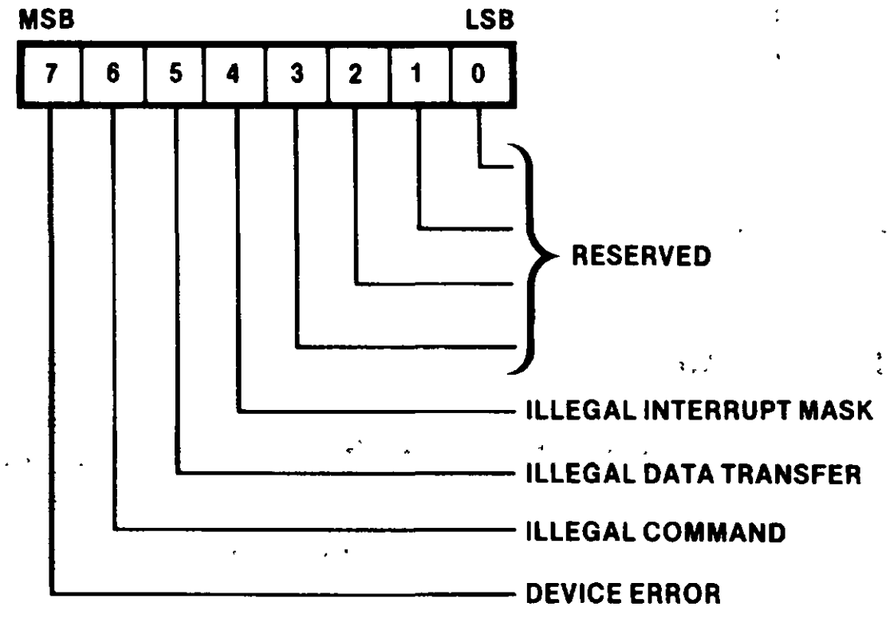
NOTE

Bit 7 of the command byte is not used in system commands (i.e., an interrupt cannot be generated by the PIO on completion of a system command).

The PACIFY command is a software reset that terminates any pending I/O opertion and reinitializes the PIO hardware and software. There is no data byte transfer associated with the PACIFY command, and a minimum of 100 milliseconds is required to complete the initialization sequence.

The ERESET command is intended for use with a peripheral device that requires a hardware error reset to clear an error condition within the device. Since the standard devices of the PIO do not require an error reset signal, the ERESET command is not implemented by PIO firmware.

The SYSTAT command causes the PIO processor to load the system status byte into the output data buffer of the DBB. The PIO processor sets that OBF flag and clears the C/$\overline{D}$ flag to inform the master processor that the system status byte can be read from the data port. The format of the system status byte is as follows:



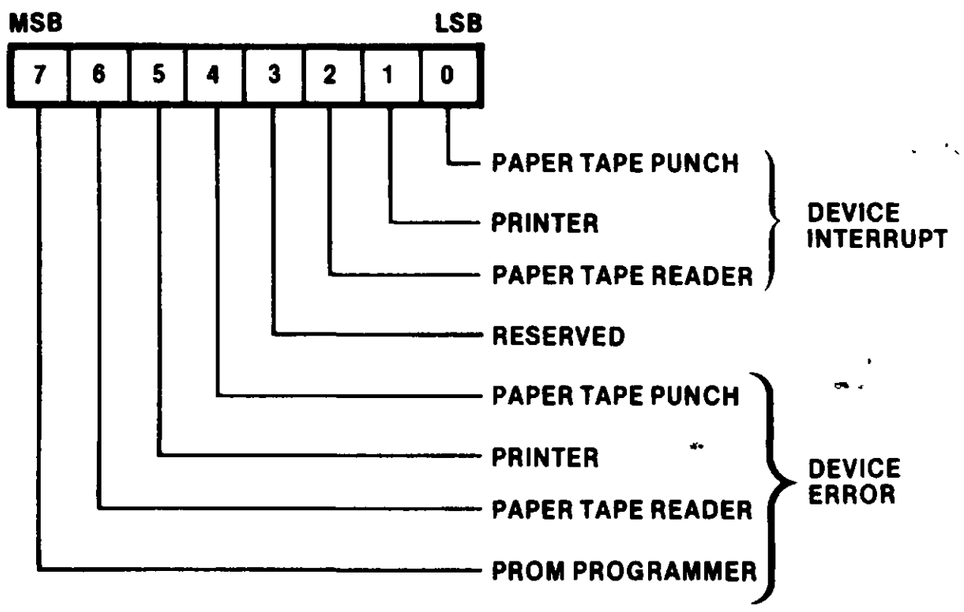
Illegal Interrupt Mask is set when the interrupt reset mask transferred by a SRQDAK command does not correspond to the interrupt bit set in the device status byte. The illegal interrupt mask bit is cleared when the master processor reads the system status byte from the DBB output buffer.

Illegal Data Transfer is set when a master processor loads a data byte into the DBB input buffer without a preceding command. The byte is not accepted by the PIO. The illegal data transfer bit is cleared when the master processor reads the system status byte from the DBB output buffer.

Illegal Command is set when a master processor loads an undefined command code into the DBB input buffer (see table 5-1). The command byte loaded is not executed by the PIO. The illegal command bit is cleared when the master processor reads the system status byte.

Device Error is set when a device fails to respond to a command. The master processor must issue a DSTAT command to determine the individual device responsible for the error and to clear the device error bit from the system status byte.

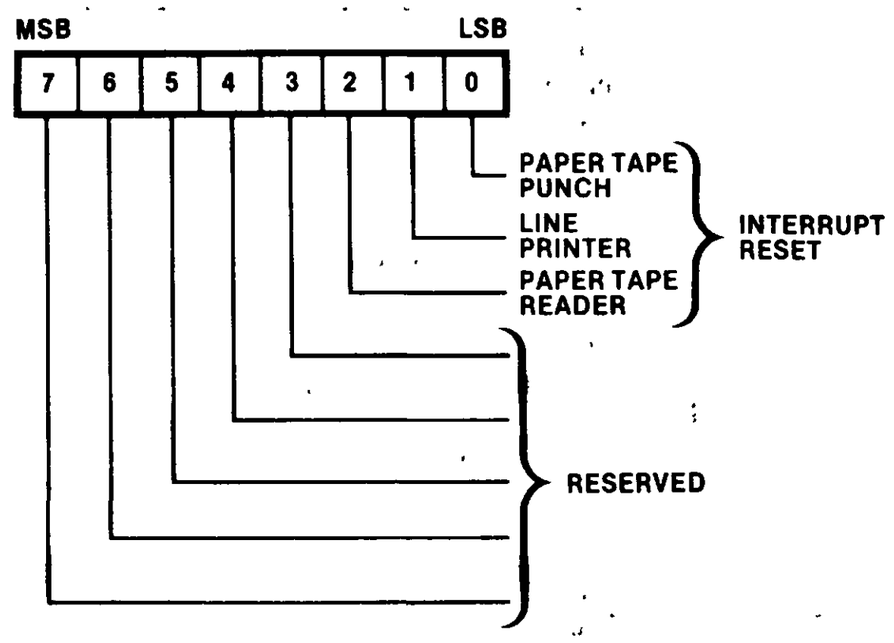
The DSTAT command causes the PIO processor to load the device status byte into the DBB output data buffer. The PIO processor sets the OBF flag and clears the C/$\overline{D}$ flag to inform the master processor that the device status byte can be read from the data port. The format of the device status byte is as follows:



A Device Interrupt bit is set when the operation specified by a command that has its request interrupt bit set has been completed. The device interrupt bit is cleared by a subsequent SRQDAK or SRQACK command.

A Device Error bit is set when the specified device is unable to comply with a command issued by the master processor. A device error bit is cleared when the master processor reads the device status byte. More detailed error information is provided by the individual device status bytes (see RSTC, PSTC and LSTC command descriptions).

The SRQDAK command is used to clear a device interrupt. The subsequent data byte from the master processor to the DBB input data buffer is as follows:



An Interrupt Reset bit, when set, clears the corresponding device interrupt bit in the device status byte (see DSTAT command description). Attempting to reset an interrupt bit that is not set causes the illegal interrupt mask bit to be set in the system status byte. Note that the SRQDAK command also clears he PIO hardware interrupt signal to the IPB/IPC.

The SRQACK command causes the PIO processor to reset all of the device interrupt bits in the device status byte and the PIO hardware interrupt signal to the IPB/IPC. A data byte transfer is not associated with the SRQACK command.

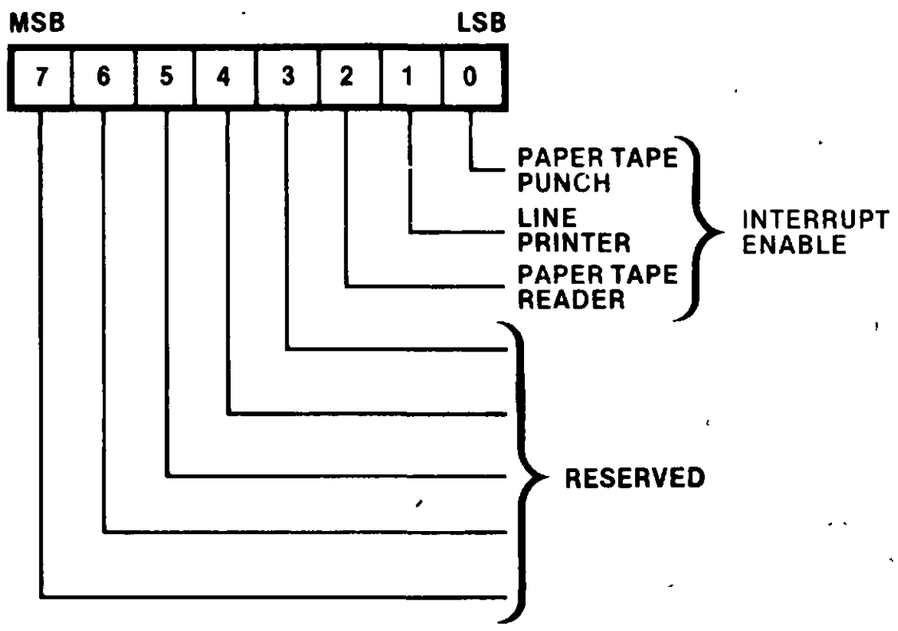
The SRQ command causes the PIO to generate a hardware interrupt to the IPB/IPC. The device interrupt bits of the device status byte are not affected and a data byte transfer is not initiated. This diagnostic command allows a master processor to test the PIO hardware interrupt signal when all other local interrupts of the IPB/IPC are reset. The PIO interrupt signal causes a level 7 interrupt request on the Multibus interface. The interrupt signal is cleared by a subsequent SRQACK command.

The DECHO command causes the PIO processor to accept and return (in complemented form) the next data byte input from the master processor. Although the data byte is sent and received via I/O port F8, the internal path within the PIO processor includes a software-controlled transfer of the data byte from the DBB input data buffer to the DBB output data buffer. The DECHO command represents a fairly comprehensive test of the master processor–PIO interface. The PIO response to a DECHO command requires approximately two milliseconds.

The CSMEM command causes the PIO processor to checksum the contents of the PIO internal ROM and thereby perform a confidence test of the PIO firmware. If the checksum test passes, the PIO processor clears the C/$\overline{D}$ flag to zero and returns a data byte of all zeroes; if the checksum test fails, the PIO processor sets the C/$\overline{D}$ flag to one and returns a data byte of all ones. Command execution requires approximately 100 milliseconds.

The TRAM command causes the PIO processor to perform read-after-write testing of PIO internal RAM. If a RAM location cannot be successfully written and read-back, the test is immediately terminated, and the PIO processor sets the C/$\overline{D}$ flag to one and returns a data byte of all ones. If the test passes, the PIO processor clears the C/$\overline{D}$ flag to zero and returns a data byte of all zeroes. Command execution requires approximately 100 milliseconds.

The SINT command causes the PIO processor to accept an interrupt enable byte at the DBB input data buffer. A bit set within the byte enables the corresponding device interrupt and also enables the PIO hardware interrupt signal to the IPB/IPC. The interrupt enable bits perform a function identical to the interrupt request bit of a command byte. Note that once a device interrupt is enabled, it remains enabled until a subsequent SINT command is issued to reset the interrupt enable bit. The format of the interrupt enable byte is as follows:



An Interrupt Enable bit, when set, enables the interrupt from the corresponding device. The format of the interrupt enable byte is identical to the interrupt reset byte of the SRQDAK command. Note that when any of the individual interrupt enable bits is set, the PIO interrupt line to the IPB/IPC is also enabled.

5.2.2 PAPER TAPE READER COMMANDS

The standard 200 character/second paper tape reader is pulsed to move one frame forward (to the right) or reverse (to the left). Two control signals are used by the reader: DR (drive right) and DL (drive left). The paper tape reader supplies two device status signals: DATA READY and SYSTEMS READY. The DATA READY signal indicates that tape movement has been completed and that a valid data byte is being presented to the PIO. The SYSTEMS READY signal indicates that the reader is connected to the PIO and that power is applied to the reader.

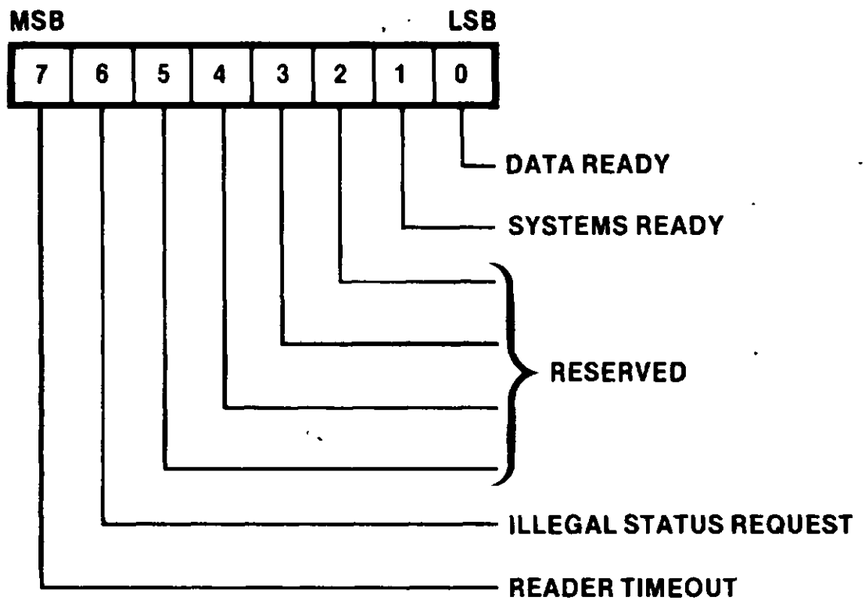
Commands to the PIO include the RDRC reader control command and the RSTC reader status command. The reader control command is able to either move the tape one frame or to enable PIO acceptance of the data byte (concurrent tape movement and data reading is not implemented). If the PIO detects an error condition during a read operation, the reader status byte is returned in place of the data byte. The PIO informs the master processor that a status byte is being presented by setting the command/data bit of the DBB status byte, and the operation is comparable to the PIO's execution of the RSTC reader status command.

The RDRC command uses bits 5 and 6 of the command byte to define the direction of tape movement, and either to select tape movement or to enable data byte reading. Note that if data byte reading is specified, the tape direction bit (bit 5) has no significance. The combinations of bits 5 and 6 are as follows:

Bit		Operation
6	5	
0	0	Read data byte
0	1	Read data byte
1	0	Move tape forward one frame
1	1	Move tape reverse one frame

When the request interrupt bit (bit 7) of the command byte is set, a PIO interrupt is generated at the completion of command processing and the paper tape reader device interrupt bit is set in the device status byte. When the RDRC command specifies tape movement (bit 6 of the command byte set), the interrupt is generated following tape movement (when the paper tape reader generates DATA READY). When the command specifies a data byte read operation, the interrupt is generated when the PIO processor places the requested data byte in the DBB output data buffer. Note that if an error is detected during command execution, the interrupt is generated when the PIO processor places the reader status byte in the DBB output data buffer.

The RSTC command causes the PIO processor to place the reader status byte in the DBB output data buffer and to clear the paper tape reader device error bit in the device status byte. The format of the reader status byte is as follows:



Data Ready is set when the paper tape reader has a character in its output buffer and is cleared when the PIO processor executes an RDRC command to access the character byte. When operating in the polled mode (interrupts disabled), a master processor tests the data ready bit following a tape movement command to determine when the character can be accessed by a subsequent RDRC read data byte command.

Systems Ready is set when both primary power is applied to the paper tape reader and the reader is connected to its corresponding connector on the rear panel of the development system.

Illegal Status Request is set when the request interrupt bit is set in the RSTC command byte (a master processor cannot legally request an interrupt at the completion of a status access operation). The illegal status request bit is cleared when the master processor reads the paper tape reader status byte from the DBB output data buffer.

Reader Timeout is set if the paper tape reader is not connected or has remained in a not ready condition (data ready=0) for more than 260 milliseconds following a tape movement command. The timer is reinitialized with every RDRC command.

5.2.3 PAPER TAPE PUNCH COMMANDS

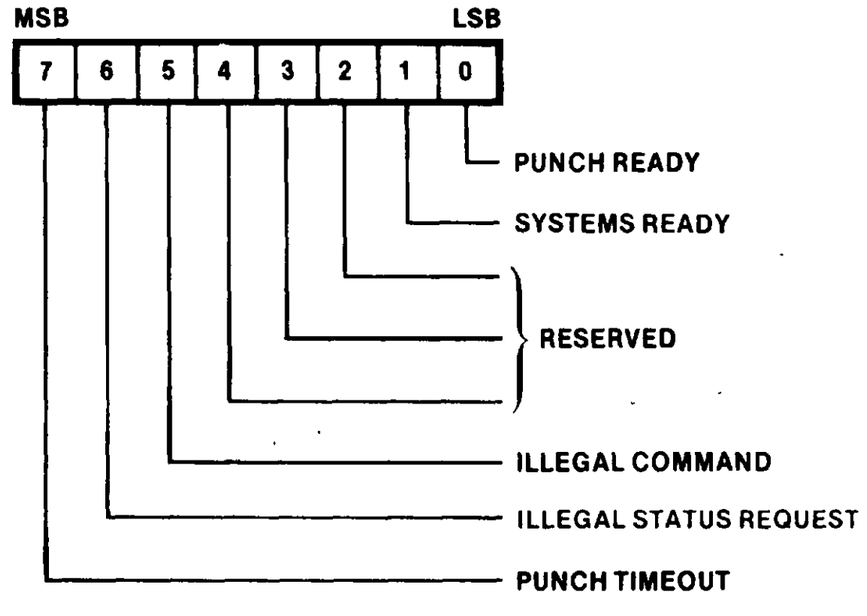
The standard 75 character/second paper tape punch responds to a single control signal (PUNCH COMMAND) that causes the punch to perforate a character on the tape and to advance the tape to the next frame position. The PUNCH COMMAND signal is generated by the PIO following receipt of a PUNC command from a master processor. The associated paper tape punch data byte is placed on the data output lines to the paper tape punch prior to the generation of the PUNCH COMMAND signal to the paper tape punch. Status regarding paper tape punch operations is supplied by the PIO processor to the master processor on receipt of a PSTC command. Status lines from the paper tape punch consist of PUNCH READY (operation complete) and SYSTEMS READY.

NOTE

The INPUT MODE SELECT and OUTPUT MODE SELECT control lines of the standard paper tape punch are disabled (grounded) at the interface connector on the chassis rear panel, and the DIRECTION control line is connected to +5 volts to only enable forward tape motion.

The PUNC command causes the PIO processor to accept the subsequent data byte at the DBB input data buffer as the character byte to be punched on the paper tape. The PIO processor delays the generation of the PUNCH COMMAND/ control signal until the character byte has stabilized on the output data lines to the paper tape punch. If the request interrupt bit of the PUNC command byte is set, the PIO processor sets the paper tape reader device interrupt bit in the device status byte and generates a PIO interrupt to the IPB/IPC when the paper tape reader returns a PUNCH READY status indication.

The PSTC command causes the PIO processor to place the paper tape punch status byte in the DBB output data buffer and to clear the paper tape punch device error bit of the device status byte. The format of the paper tape punch status byte is as follows:



Punch Ready is set when the paper tape punch is ready to receive a new punch command (i.e., after a character is punched and the tape is advanced one frame). When operating in the polled mode (interrupts disabled), a master processor tests the punch ready bit following a PUNC command to determine when the operation has been completed and the next character can be punched.

Systems Ready is set when both primary power is applied to the paper tape punch and the punch is connected to its corresponding connector on the development system's rear panel.

Illegal Command is set when a PUNC command is followed by a subsequent command byte ($C/\overline{D}=1$). When this bit is set, the PUNC command is not executed (a character is not punched). The illegal command bit is cleared when the master processor reads the paper tape punch status byte from the DBB output data buffer.

Illegal Status Request is set when the request interrupt bit is set in the PSTC command byte (a master processor cannot legally request an interrupt at the completion of a status access operation). The illegal status request bit is cleared when the master processor reads the paper tape punch status byte from the DBB output data buffer.

Punch Timeout is set if the paper tape punch is not connected or has remained in a not ready condition (punch ready=0) for more than 260 milliseconds following a PUNC command. The timer is reinitialized with every PUNC command.

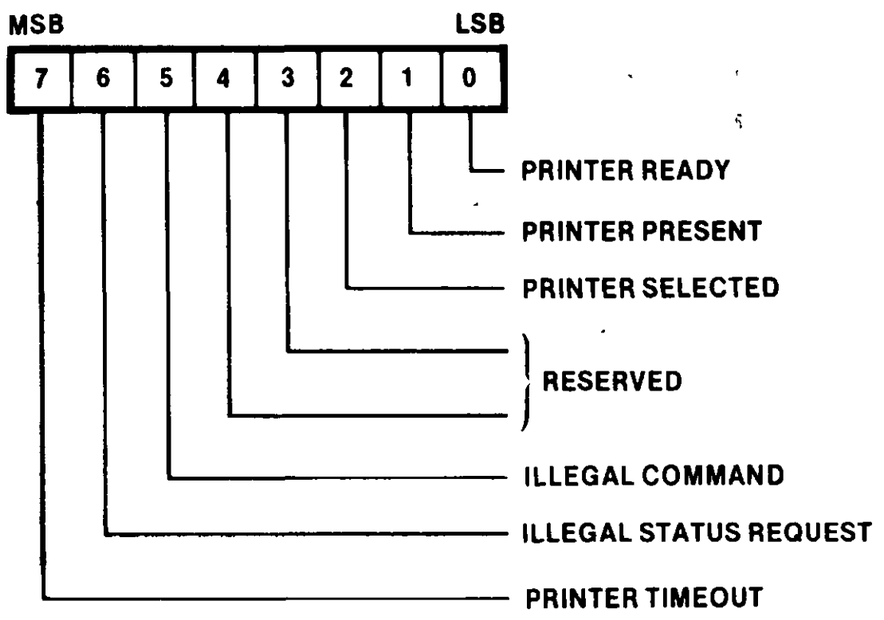
5.2.4 PRINTER COMMANDS

Two PIO commands are used to control the standard 60 character per second dot matrix printer. One command transfers ASCII character and control codes from a master processor to the line printer (both character and control codes are output to the printer as parallel bytes), and the other command returns the printer status byte to the master processor. The PIO processor uses a single control line (LPT DATA STROBE/) to transfer both control and character codes to the printer. The ASCII characters transferred are stored in the printer's input character buffer until either 80 characters have been received or a carriage return or line feed control code is encountered to terminate the line. When a character line is complete, the printer initiates its print cycle.

The line printer returns three status signals to the PIO processor: SELECT, BUSY and ACKNOWLEDGE. The SELECT signal is active when the line printer is placed on-line, and the BUSY signal is active during a print cycle. The ACKNOWLEDGE signal, when active, indicates that a character or control code has been accepted. The duration of the ACKNOWLEDGE signal returned by the standard printer is too short to be detected by the PIO processor; the PIO processor assumes that all character and control codes are accepted.

The LPTC command causes the PIO processor to accept the subsequent data byte at the DBB input data buffer as an ASCII character or control code to the line printer. The PIO processor delays the generation of the data strobe pulse (LPT DATA STROBE/) to the line printer until the ASCII code byte has stabilized on the output data lines to the printer. If the request interrupt bit (bit 7) of the LPTC command byte is set, the PIO processor sets the printer device interrupt bit in the device status byte and generates a PIO interrupt to the IPB/IPC when the associated character line is printed (i.e., when BUSY returns to an inactive level).

The LSTC command causes the PIO processor to place the printer status byte in the DBB output data buffer and to clear the printer device error bit in the device status byte. The format of the printer status byte is as follows:



Printer Ready is set whenever the printer is on-line and is not performing a print cycle (i.e., when BUSY is inactive). Since a character or control code is not accepted during a print cycle, the printer ready bit is examined prior to every LPTC command.

Printer Present is set when the printer is connected, powered-on and has been placed off-line, the printer present bit remains active.

Printer Selected is set when the printer is placed on-line (i.e., when SELECT is active) and remains set until the printer is placed off-line.

Illegal Command is set when a LPTC command is followed by a subsequent command byte (C/$\overline{D}$=1). When the illegal command bit is set, the LPTC command is not executed (a data strobe pulse is not generated). The illegal command bit is cleared when the master processor reads the printer status byte from the DBB output data buffer.

Illegal Status Request is set when the request interrupt bit is set in the LSTC command byte (a master processor cannot legally request an interrupt at the completion of a status access operation). The illegal status request bit is cleared when the master processor reads the printer status byte from the DBB output data buffer.

Printer Timeout is set when the printer fails to complete a print cycle within 3.5 seconds. The cycle timer is reinitialized at the beginning of every print cycle when BUSY goes active.

5.2.5 PROM PROGRAMMER COMMANDS

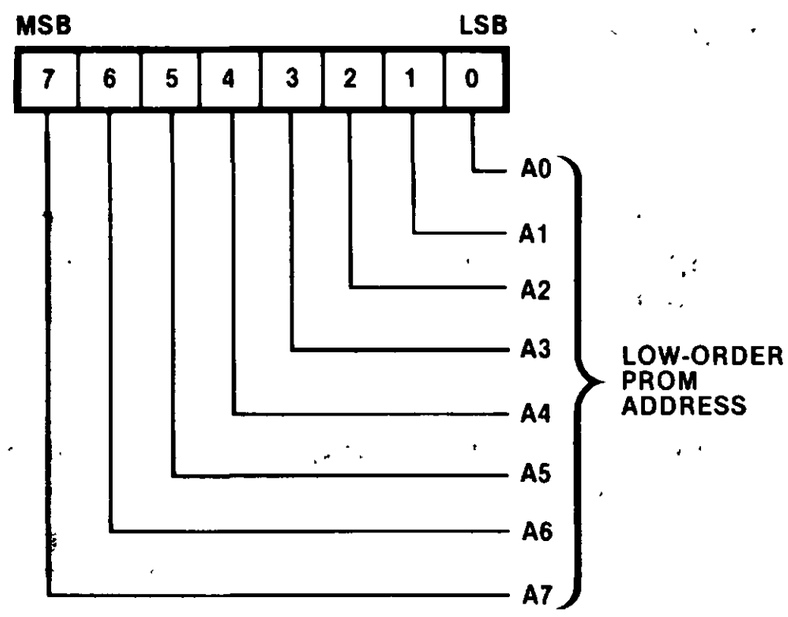
The following information pertains to the PROM programmer interface of the Intellec Series II development system. The text provides details on each of the commands and describes the functions of bits within the currently implemented data and status bytes. The PIO processor does not examine the contents of any PROM programmer data byte.

One factor common to all PROM programmer commands is that the PIO processor initiates a three millisecond timeout each time a command is received. For the WPPC write command, the timeout error simply causes a PROM programmer timeout bit to be set in the device status byte. However, for the RPPC and RDPDC read commands and the RPSTC status command, a special action is taken since it is assumed that a valid data or status byte could not be accessed from the PROM programmer. This special action consists of setting the C/$\overline{D}$ flag of the DBB status byte and loading an all ones byte (FFH) into the DBB output data buffer. The master program recognizes this DBB response as an indication that the PROM programmer was unable to provide the requested data or status byte.

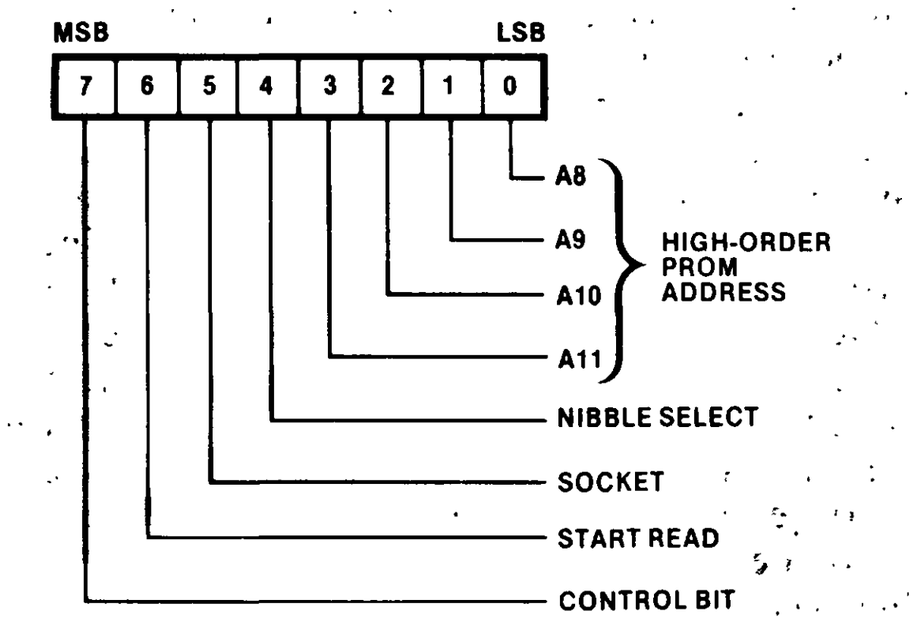
The WPPC command causes the PIO to transfer three bytes in sequence from the DBB input data buffer to the PROM programmer interface write data lines (PWD0-PWD7). Concurrently, the PIO processor sequentially activates control lines PPWC2, PPWC1, and PPWC0 in order to synchronize acceptance of the three bytes by the PROM programmer. The PROM programmer responds to each control signal by generating PPACK (PROM Programmer Acknowledge) to acknowledge acceptance of the associated byte. PPACK, since it cannot be detected at the PIO processor's data port due to its short duration, is additionally coupled to the TEST 1 input where it is tested using the appropriate branch instruction.

The three bytes transferred to the PROM programmer by the WPPC command are described in the following text in the order of their occurrence.

The first of the three write bytes is clocked by PPWC2 and is known as the low address byte. The format of the low address byte is as follows:



The second of three write bytes is clocked by PPWC1 and is known as the high address byte. The format of the high address byte is as follows:



High-Order PROM Address is defined by the control bit (bit 7) of the high address byte. When the control bit is clear, the four low-order bits of the byte contain PROM address bits A8 through A11 (4k address range). When the control bit is set, the four low-order bits contain PROM address bits A12 through A15 (64k maximum address range).

Nibble Select is only applicable when programming 4-bit PROMs. When the nibble select bit is set, the four data bits to be programmed are contained in the high-order four bits of the data byte (D7-D4), and when clear, the four data bits are contained in the low-order four bits of the data byte (D3-D0). The nibble select bit is ignored when programming 8-bit PROMs. When reading a 4-bit PROM, the data is always returned in the four low-order bits of the data byte.

Socket Select, when set, specifies PROM programming operations on the PROM device installed in Socket 1 and, when clear, specifies PROM programming operations on the PROM device in Socket 2.

Start Read is only applicable during PROM programmer read operations (see RPPC and RDPDC read command descriptions).

Control Bit defines the four address bits of the high address byte. When set, the high-order PROM address bits correspond to address bits A12 through A15 and when clear, the high-order PROM address bits correspond to address bits A8 through A11. Note that since the PIO firmware only issues one high address byte with any read or write command, when address bits A12 through A15 are required (e.g., when programming a 2764 EPROM), the master processor must perform a "dummy read" (with the control bit set) to load address bits A12 through A15 prior to programming (or reading) the PROM.

The third of the three write bytes is clocked by PPWC0 and is the data byte to be written into the PROM address specified by the low and high address bytes.

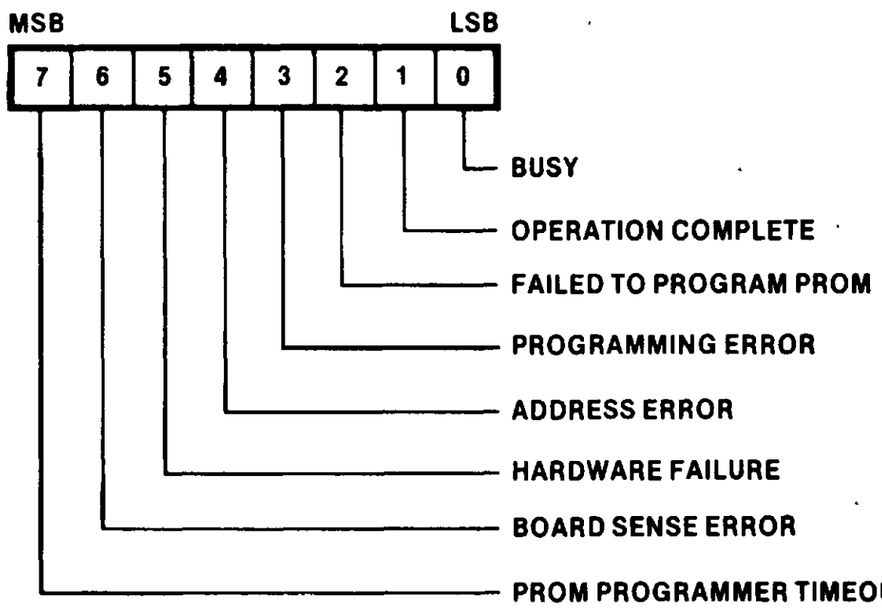
The RPPC command is used to read the contents of a PROM that has been previously programmed. The action of the PIO processor in response to an RPPC command is similar to that of the WPPC command in that the low and high address bytes are transferred to the PROM programmer by sequential generation of PPWC2 and PPWC1. If the start read bit of the high address byte is clear, the PIO processor generates PPRC0 in place of PPWC0 to cause the PROM programmer to transfer the data contents of the addressed PROM location to the PIO processor's input data port. The PROM programmer signals the PIO processor when the data byte is available (PPACK), and the PIO processor sets the OBF flag in the DBB status byte when it transfers the byte to the DBB output data buffer for access by the master processor.

If the start read bit of the high address byte is set, the "read polling mode" is selected, and the PROM programmer initiates the read operation when the high address byte is received (i.e., on receipt of PPWC1). Note that in the read polling mode, the low address byte must precede the high address byte. The master processor monitors the Busy bit in the operation status byte (using the RPSTC command) to determine when the PROM programmer has accessed the addressed data byte. When the byte has been accessed (when the Busy bit clears), the master processor must issue an RDPDC command to access the data byte. Note that if the data byte returned by an RPPC or RDPDC command is invalid, the PIO processor sets the command/data flag ($C/\overline{D}=1$) and returns a data byte of all ones.

The RDPDC command is only valid when a prior RPPC command specifies read polling mode operation (start read bit set in the high address byte). The RDPDC command, as previously explained, is issued by the master processor when it determines that the PROM programmer has accessed the data byte addressed by the RPPC command. In response to the RDPDC command, the PIO processor generates PPRC0 to enable the byte onto the PROM programmer's read data lines (the byte is accepted on receipt of PPACK from the PROM programmer) and then transfers the byte from its input data port to the DBB output data buffer for access by the master processor.

The RPSTC command causes the PIO processor to transfer two status bytes to the master processor. On receipt of the RPSTC command, the PIO processor generates PPRC1 to access the operation status byte from the PROM programmer. The accessing of the operation status byte is simplified in that this byte is continuously applied to the read data lines (PRD0-PRD7) except when the PROM programmer receives a read data command. On receipt of PPACK from the PROM programmer, the PIO processor transfers the operation status byte to the DBB output data buffer and then monitors the DBB output buffer full (OBF) flag to determine when the master processor accepts the status byte. When the byte is accepted (when the OBF flag is cleared), the PIO processor places the PROM programmer device status byte in the DBB output data buffer and again sets the OBF flag.

The format of the PROM programmer operation status byte is as follows:



Busy is set while the PROM programmer is performing any operation specified by a data read or write command. Note that the remaining status bits are only valid when the PROM programmer is *not* busy (i.e., when the busy bit is clear).

Operation Complete is set at the completion of any data read or write operation.

Failed to Program PROM is set when the PROM programmer is unable to successfully program the PROM.

Programming Error is set when an attempt is made to reprogram a fused location (e.g., attempting to reprogram a bipolar PROM).

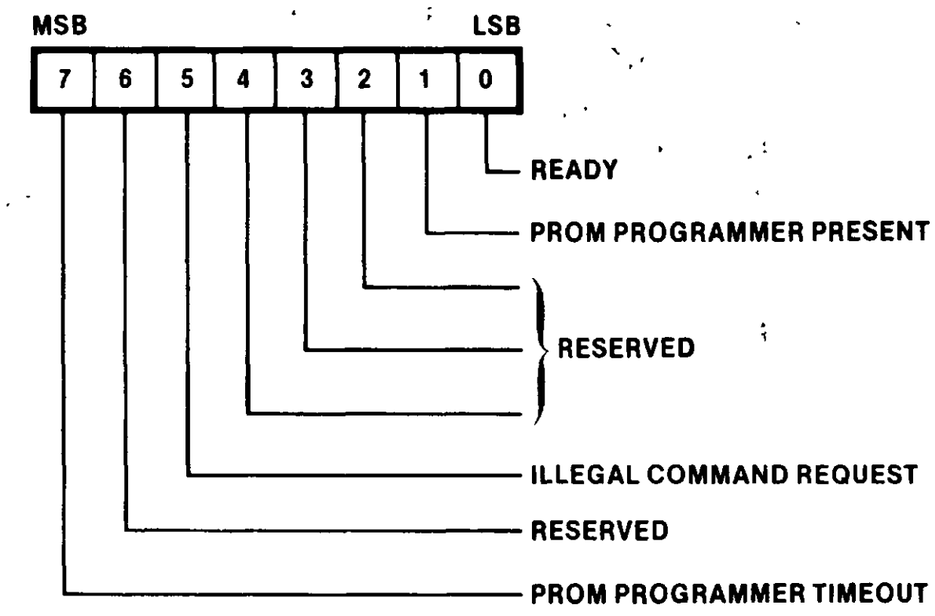
Address Error is set when an attempt is made to program or read a non-existent PROM location.

Hardware Failure is set when a hardware fault is detected (e.g., programming voltage not present, PROM incorrectly programmed).

Board Sense Error is set when a personality module is not installed for the socket specified.

PROM Programmer Timeout is a personality module-dependent status bit (e.g., the UPP-848 personality module sets this bit if the 8748 device is incorrectly oriented in the socket).

The format of the PROM programmer device status byte is as follows:



Ready is set when the PIO processor completes execution of a command.

PROM Programmer Present is set when the PROM programmer is connected to its corresponding rear panel connector and primary power is applied to the PROM programmer.

Illegal Command Request is set when the master processor issues a new command and the previous command has not been completed.

PROM Programmer Timeout is set if the PROM Programmer is not present or primary power is not enabled, or if command execution is not completed within three milliseconds (i.e., the ready bit remains clear for more than three milliseconds).

```
ISIS-II 8008/8085 MACRO ASSEMBLER, V2.0        KEYINT   PAGE    1

 LOC  OBJ          SEQ          SOURCE STATEMENT

                      1           NAME    KEYINT
                      2           ASEG
                      3 ;
                      4 ;         THIS PROGRAM IS AN EXAMPLE OF AN INTERRUPT ROUTINE TO BE USED
                      5 ;         WHEN ENABLING AND SERVICING INTERRUPTS ORIGINATING FROM THE DEVICES
                      6 ;         ATTACHED TO THE LOCAL INTERRUPT CONTROLLER (IOC,PIO,USART1,USART2 ETC)
                      7 ;         OF A SERIES II PROCESSOR.
                      8
                      9 ;         THIS SPECIFIC EXAMPLE IS FOR A KEYBOARD INTERRUPT FROM THE SERIES II
                     10 ;         MODEL 220 OR 230. THE LINES OF CODE WHICH DO NOT CHANGE FROM DEVICE TO DEVICE
                     11 ;         ARE SHOWN WITH AN ASTERIK(*). THE LINES WHICH CHANGE BASED ON THE
                     12 ;         DEVICE IS SHOWN WITH A DOUBLE ASTERIK(**).
                     13 ;
                     14 ;         THE FOLLOWING STEPS ARE PERFORMED
                     15 ;                 INITIALIZATION:
                     16 ;                  1) UNMASK INTERRUPT LEVEL 0,7 OF SYSTEM 8259
                     17 ;                  2) UNMASK INTERRUPT LEVEL 6 OF LOCAL 8259
                     18 ;                  3) ENABLE KEYBOARD INTERRUPT OF IOC
                     19 ;
                     20 ;                 INTERRUPT SERVICING:
                     21 ;                  1) POLL LOCAL 8259 (MUST BE DONE)
                     22 ;                  2) READ CHARACTER FROM IOC KEYBOARD
                     23 ;                  3) ACKNOWLEDGE INTERRUPT TO IOC LOGIC
                     24 ;                  4) ACKNOWLEDGE INTERRUPT (EOI) TO SYSTEM/LOCAL 8259
                     25 ;
                     26 ;         THE ROUTINE WHEN STARTED FROM LOCATION 0100H ENABLES KEYBOARD INTERRUPTS
                     27 ;         AND ECHOS THE KEYBOARD CHARACTER TO THE CONSOLE DURING THE SERVICE
                     28 ;         ROUTINE FOLLOWED BY RETURNING TO THE EXECUTING ROUTINE WHICH
                     29 ;         IN THIS CASE IS A LOOP ON A HALT INSTRUCTION.
                     30 ;
                     31 ;         8259 INTERRUPT CONTROLLER INTERFACE CONSTANTS
                     32 ;
                     33 ;
00FA                 34 LICP0    EQU     0FAH            ; LOCAL INTERRUPT CONTROL PORT 0
00F8                 35 LICP1    EQU     0F8H            ; LOCAL INTERRUPT CONTROL/DATA PORT 1
00FC                 36 SICP0    EQU     0FCH            ; SYSTEM INTERRUPT CONTROL PORT 0
00FD                 37 SICP1    EQU     0FDH            ; SYSTEM INTERRUPT CONTROL/DATA PORT 1
                     38 ;
                     39 ;
                     40 ;
0020                 41 EOI      EQU     020H            ; END OF INTERRUPT
00C0                 42 OCW      EQU     00CH            ; OPERATION CONTROL WORD (SOLL COMMAND)
                     43 ;
                     44 ;         SYSTEM INTERRUPT CONSTANTS
                     45 ;
007E                 46 SIMSK    EQU     07EH            ; MASK FOR ENABLING SYSTEM INTERRUPTS 0,7
00BF                 47 IOCIM    EQU     0BFH            ; MASK FOR ENABLING IOC INTERRUPT
                     48 ;
                     49 ;         IOC INTERFACE CONSTANTS
                     50 ;
0004                 51 IACK     EQU     004H            ; INTERRUPT ACKNOWLEDGE COMMAND
000A                 52 SINT     EQU     00AH            ; SYSTEM INTERRUPT ENABLE/DISABLE COMMMAND
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0B12                53 KEYD     EQU     0B12H           ; KEYBOARD DATA COMMAND
0B82                54 KMSK     EQU     0B82H           ; KEYBOARD MASK
                    55 ;
                    56 ;        MONITOR INTERFACE CONSTANTS
                    57 ;
F889                58 CO       EQU     0F889H          ; CONSOLE OUT ENTRY POINT
F821                59 IOCDR1   EQU     0F821H          ; IOC OUTPUT DRIVER
F844                60 IOCDR2   EQU     0F844H          ; IOC INPUT DRIVER
                    61 ;
                    62 ;
                    63 ;        INTERRUPT 7 ROUTINE
                    64 ;
0B38                65          ORG     0B38H
0B38 3E0C           66          MVI     A,OCW           ; ** OUTPUT POLL COMMAND TO LOCAL 8259
0B3A D3F8           67          OUT     LICP1           ; **
0B3C DBF8           68          IN      LICP1           ; ** INPUT HIGHEST PRIORITY INTERRUPT AND
                    69                                  ; DISCARD IF DESIRED
0B3E 0612           70          MVI     B,KEYD          ; * LOAD IOC KEYBOARD COMMAND
0B40 CD21F8         71          CALL    IOCDR1          ; * INPUT KEYBOARD DATA
0B43 4F             72          MOV     C,A             ; SET UP FOR OUTPUT TO CONSOLE ROUTINE
0B44 CD89F8         73          CALL    CO              ; OUTPUT CHARACTER
0B47 0604           74          MVI     B,IACK          ; * LOAD IOC INTERRUPT ACKNOWLEDGE COMMAND
0B49 0E82           75          MVI     C,KMSK          ; * LOAD KEYBOARD ACKNOWLEDGE CONSTANT
0B4B CD44F8         76          CALL    IOCDR2          ; * OUTPUT ACKNOWLEDGE
0B4E 3E20           77          MVI     A,EOI           ; ** OUTPUT EOI TO LOCAL/SYSTEM INTERRUPT CONTROLLERS
0B50 D3F8           78          OUT     LICP1           ; **
0B52 D3FD           79          OUT     SICP1           ; **
0B54 FB             80          EI                      ; ** ENABLE INTERRUPTS
0B55 C9             81          RET                     ; **
                    82 ;
                    83 ;        MAIN PROGRAM
                    84 ;
0B00                85          ORG     0B00H
0B00 3E7E           86          MVI     A,SIMSK         ; ** OUTPUT SYSTEM INTERRUPT MASK
0B02 D3FC           87          OUT     SICP0           ; **
0B04 3EBF           88          MVI     A,IOCIM         ; ** OUTPUT LOCAL INTERRUPT MASK
0B06 D3FA           89          OUT     LICP0           ; **
0B08 060A           90          MVI     B,SINT          ; * LOAD SYSTEM INTERRUPT COMMAND
0B0A 0E82           91          MVI     C,KMSK          ; * LOAD KEYBOARD ENABLE CONSTANT
0B0C CD44F8         92          CALL    IOCDR2          ; * OUTPUT TO IOC LOGIC
                    93 LOOP:
0B0F 76             94          HLT
0B10 C30F0B         95          JMP     LOOP
                    96          END

PUBLIC SYMBOLS


EXTERNAL SYMBOLS


USER SYMBOLS
CO      A F889    EOI     A 0020    IACK    A 0004    IOCDR1  A F821    IOCDR2  A F844    IOCIM   A 00BF    KEYD    A 0B12
KMSK    A 0082    LICP0   A 00FA    LICP1   A 00F8    LOOP    A 0B0F    OCW     A 000C    SICP0   A 00FC    SICP1   A 00FD
SIMSK   A 007E    SINT    A 000A
```

```
ASM80 :F1:IDRVR.SRC OBJECT(:F1:IDRVR.OBJ) PRINT(:F1:IDRVR.LST)

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    1

  LOC  OBJ          SEQ          SOURCE STATEMENT

  4000                1          ORG     4000H           ; Arbitrary origin point
  4000 C33040         2          JMP     START           ; Branch to start of example
                      3 ;-----------------------------------------------------------------
                      4 ;
                      5 ; BASIC INPUT DRIVER EXAMPLE
                      6 ;
                      7 ; This is an example of an input driver to the IOC.  It is called
                      8 ; with the following input parameters:
                      9 ;           D-register contains the (device status command)
                     10 ;           E-register contains the (device output command)
                     11 ; Upon exiting from this driver, the data input from the device
                     12 ; will be in the A-register.
                     13 ;
                     14 ; An input driver to the PIO is similar to this driver to the IOC
                     15 ; except that the I/O port assignments are changed.
                     16 ;
                     17 ;-----------------------------------------------------------------
                     18 ;
  000D               19 DISABL  EQU     0DH             ; Disable interrupts
  0005               20 ENABL   EQU     05H             ; Enable interrupts
  00FF               21 CPUC    EQU     0FFH            ; Control port
  00C0               22 IOCI    EQU     0C0H            ; I/O Controller input data (from DBB) port
  00C0               23 IOCO    EQU     0C0H            ; I/O Controller output data (to DBB) port
  00C1               24 IOCS    EQU     0C1H            ; I/O Controller input DBB status port
  00C1               25 IOCC    EQU     0C1H            ; I/O Controller output control command port
  0004               26 F0      EQU     00000100B       ; Flag 0 - slave is busy, master is locked out
  0002               27 IBF     EQU     00000010B       ; Slave input buffer is full
  0001               28 OBF     EQU     00000001B       ; Slave output buffer is full
  0001               29 DEVRDY  EQU     00000001B       ; Device ready
                     30 INDRVR:
  4003 3E0D          31         MVI     A,DISABL        ; Block all interrupts
  4005 D3FF          32         OUT     CPUC
                     33 LOOP1:
  4007 DBC1          34         IN      IOCS            ; Input DBB status
  4009 E607          35         ANI     F0 OR IBF OR OBF; Test for slave processor idle
  400B C20740        36         JNZ     LOOP1           ; Loop until it is idle
  400E 7A            37         MOV     A,D             ; Load the device status command
  400F D3C1          38         OUT     IOCC            ; Output the command to the IOC control port
                     39 LOOP2:
  4011 DBC1          40         IN      IOCS            ; Input DBB status
  4013 E607          41         ANI     F0 OR IBF OR OBF; Mask off status flags
  4015 FE01          42         CPI     OBF             ; Test for slave done; something for the master
  4017 C21140        43         JNZ     LOOP2           ; Loop until slave is ready
  401A DBC0          44         IN      IOCI            ; Otherwise device from DBB
  401C E601          45         ANI     DEVRDY          ; Is the device ready?
  401E CA0740        46         JZ      LOOP1           ; Loop until it is
                     47 LOOP3:
  4021 DBC1          48         IN      IOCS            ; Input DBB status
  4023 E607          49         ANI     F0 OR IBF OR OBF; Test for slave processor idle
  4025 C22140        50         JNZ     LOOP3           ; Loop until it is idle
  4028 78            51         MOV     A,E             ; Load the device output command
  4029 D3C1          52         OUT     IOCC            ; Output the command to the PIO control port
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    2

  LOC  OBJ          SEQ          SOURCE STATEMENT

                     53 LOOP4:
  402B DBC1          54           IN      IOCS              ; Input DBB status
  402D E607          55           ANI     FO OR IBF OR OBF; Mask off status flags
  402F FE01          56           CPI     OBF               ; Test for slave done; something for the master
  4031 C22B40        57           JNZ     LOOP4             ; Loop until it is ready
  4034 DBC0          58           IN      IOCI              ; Otherwise input the data from the DBB
  4036 F5            59           PUSH    PSW               ; Save the data
  4037 3E05          60           MVI     A,ENABL           ; Enable the interrupts
  4039 D3FF          61           OUT     CPUC
  403B F1            62           POP     PSW               ; Return the data in the A-register
  403C C9            63           RET
                     64 ;-------------------------------------------------------------------------
                     65 ;
                     66 ; This is an example keyboard driver which calls the above
                     67 ; generalized input driver
                     68 ; Characters are input from the keyboard and stored into a
                     69 ; buffer of size 122.  Input is halted when either a carriage
                     70 ; return is input or the buffer is full
                     71 ;
                     72 ;-------------------------------------------------------------------------
                     73 ;
  0012               74 KEYC    EQU     012H              ; Keyboard input data command
  0013               75 KSTS    EQU     013H              ; Keyboard device status command
                     76 START:
  403D 215940        77           LXI     H,BUFFER
  4040 067A          78           MVI     B,122             ; Set up buffer size counter of 122
                     79 LOOP:
  4042 1613          80           MVI     D,KSTS            ; Load keyboard status command
  4044 1E12          81           MVI     E,KEYC            ; Load keyboard input data command
  4046 CD0340        82           CALL    INDRVR            ; Call the input driver routine
  4049 77            83           MOV     M,A               ; Store the input byte in the buffer
  404A FE0D          84           CPI     0DH               ; Is it a carriage return?
  404C CA5840        85           JZ      EXIT              ; Branch to EXIT if we have just input a carriage return
  404F 23            86           INX     H                 ; Otherwise move the buffer pointer
  4050 05            87           DCR     B                 ;    and decrement the buffer count
  4051 CA5740        88           JZ      OVFL              ; If buffer full, jump to overflow code
  4054 C34240        89           JMP     LOOP              ; Otherwise get the next character
                     90 OVFL:
  4057 76            91           HLT                       ; Overflow code
                     92 EXIT:
  4058 76            93           HLT
  007A               94 BUFFER:   DS      122
  405D               95           END     START

PUBLIC SYMBOLS


EXTERNAL SYMBOLS


USER SYMBOLS
BUFFER A 4059    CPUC   A 00FF    DEVRDY A 0001    DISABL A 000D    ENABL  A 0005    EXIT   A 4058    FO     A 0004
IBF    A 0002    INDRVR A 4003    IOCC   A 00C1    IOCI   A 00C0    IOCO   A 00C0    IOCS   A 00C1    KEYC   A 0012
KSTS   A 0013    LOOP   A 4042    LOOP1  A 4007    LOOP2  A 4011    LOOP3  A 4021    LOOP4  A 402B    OBF    A 0001
OVFL   A 4057    START  A 403D

ASSEMBLY COMPLETE,  NO ERRORS
```

```
ASM80 :F1:ODRVR.SRC OBJECT(:F1:ODRVR.OBJ) PRINT(:F1:ODRVR.LST)

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    1

  LOC  OBJ          SEQ          SOURCE STATEMENT

  4100                1           ORG     4100H           ; Arbitrary origin point
  4100 C33A41         2           JMP     START           ; Branch to start of example
                      3 ;-----------------------------------------------------------------
                      4 ;
                      5 ; BASIC OUTPUT DRIVER EXAMPLE
                      6 ;
                      7 ; This is an example of an output driver to the PIO.  It is called
                      8 ; with the following input parameters:
                      9 ;             C-register contains the byte data to be output
                     10 ;             D-register contains the <device status command>
                     11 ;             E-register contains the <device output command>
                     12 ;
                     13 ; An output driver to the IOC is similar to this driver to the PIO
                     14 ; except that the I/O port assignments are changed.
                     15 ;
                     16 ;-----------------------------------------------------------------
                     17 ;
  000D               18 DISABL   EQU     0DH             ; Disable interrupts
  0005               19 ENABL    EQU     05H             ; Enable interrupts
  00FF               20 CPUC     EQU     0FFH            ; Control port
  00F8               21 PIOI     EQU     0F8H            ; Parallel I/O input data (from DBB) port
  00F8               22 PIOO     EQU     0F8H            ; Parallel I/O output data (to DBB) port
  00F9               23 PIOS     EQU     0F9H            ; Parallel I/O input DBB status port
  00F9               24 PIOC     EQU     0F9H            ; Parallel I/O output control command port
  0004               25 F0       EQU     00000100B       ; Flag 0 - slave is busy, master is locked out
  0002               26 IBF      EQU     00000010B       ; Slave input buffer is full
  0001               27 OBF      EQU     00000001B       ; Slave output buffer is full
  0001               28 DEVRDY   EQU     00000001B       ; Device ready
                     29 OUTDVR:
  4103 3E0D          30          MVI     A,DISABL        ; Block all interrupts
  4105 D3FF          31          OUT     CPUC
                     32 LOOP1:
  4107 DBF9          33          IN      PIOS            ; Input DBB status
  4109 E607          34          ANI     F0 OR IBF OR OBF; Test for slave processor idle
  410B C20741        35          JNZ     LOOP1           ; Loop until it is idle
  410E 7A            36          MOV     A,D             ; Load the device status command
  410F D3F9          37          OUT     PIOC            ; Output the command to the PIO control port
                     38 LOOP2:
  4111 DBF9          39          IN      PIOS            ; Input DBB status
  4113 E607          40          ANI     F0 OR IBF OR OBF; Mask off status flags
  4115 FE01          41          CPI     OBF             ; Test for slave done; something for the master
  4117 C21141        42          JNZ     LOOP2           ; Loop until slave is ready
  411A DBF8          43          IN      PIOI            ; Otherwise device from DBB
  411C E601          44          ANI     DEVRDY          ; Is the device ready?
  411E CA0741        45          JZ      LOOP1           ; Loop until it is
                     46 LOOP3:
  4121 DBF9          47          IN      PIOS            ; Input DBB status
  4123 E607          48          ANI     F0 OR IBF OR OBF; Test for slave processor idle
  4125 C22141        49          JNZ     LOOP3           ; Loop until it is idle
  4128 7B            50          MOV     A,E             ; Load the device output command
  4129 D3F9          51          OUT     PIOC            ; Output the command to the PIO control port
                     52 LOOP4:
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    2

  LOC  OBJ          SEQ          SOURCE STATEMENT

  4128 DBF9           53          IN      PIOS            ; Input DBB status
  412D E607           54          ANI     F0 OR IBF OR OBF; Test for slave processor ready
  412F C22B41         55          JNZ     LOOP4           ; Loop until it is ready
  4132 79             56          MOV     A,C             ; Load data to be written
  4133 D3F8           57          OUT     PIOD            ; Output data to the DBB
  4135 3E05           58          MVI     A,ENABL         ; Enable the interrupts
  4137 D3FF           59          OUT     CPUC
  4139 C9             60          RET
                      61 ;-------------------------------------------------------------------------
                      62 ;
                      63 ; This is an example line printer driver which calls the above generalized
                      64 ; output driver.
                      65 ; The contents of the array BUFFER are output to the line printer
                      66 ;
                      67 ;-------------------------------------------------------------------------
                      68 ;
  0003                69 ETX     EQU     03H             ; End-of-file
  0014                70 LPTC    EQU     014H            ; Line printer output data command
  0015                71 LSTC    EQU     015H            ; Line printer device status command
                      72 START:
  413A 215041         73          LXI     H,BUFFER
                      74 LOOP:
  413D 4E             75          MOV     C,M             ; Move data to be output to the C-register
  413E 1615           76          MVI     D,LSTC          ; Load printer status command
  4140 1E14           77          MVI     E,LPTC          ; Load printer output data command
  4142 CD0341         78          CALL    OUTDVR          ; Call the output driver routine
  4145 79             79          MOV     A,C
  4146 FE03           80          CPI     ETX
  4148 CA4F41         81          JZ      EXIT            ; Branch to EXIT if we have just output an ETX
  414B 23             82          INX     H
  414C C33D41         83          JMP     LOOP
                      84 EXIT:
  414F 76             85          HLT
  4150 4C494E45       86 BUFFER: DB       'LINE PRINTER PIO EXAMPLE '.0DH,0AH,ETX
  4154 20505249
  4158 4E544552
  415C 2050494F
  4160 20455841
  4164 4D504C45
  4168 20
  4169 0D
  416A 0A
  416B 03
  413A                87          END     START

PUBLIC SYMBOLS


EXTERNAL SYMBOLS


USER SYMBOLS
BUFFER A 4150    CPUC   A 00FF    DEVRDY A 0001    DISABL A 0000    ENABL  A 0005    ETX    A 0003    EXIT   A 414F
F0     A 0004    IBF    A 0002    LOOP   A 413D    LOOP1  A 4140    LOOP2  A 4141    LOOP3  A 4121    LOOP4  A 412B
LPTC   A 0014    LSTC   A 0015    OBF    A 0001    OUTDVR A 4103    PIOC   A 00F9    PIOI   A 00F8    PIOO   A 00F8
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    3

PIOS    A 00F9    START  A 413A

ASSEMBLY COMPLETE,  NO ERRORS
```

```
ASM80 :F1:DISK.SRC OBJECT(:F1:DISK.OBJ) PRINT(:F1:DISK.LST)

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    1

  LOC  OBJ          SEQ          SOURCE STATEMENT

                      1 ;*************************************************************
                      2 ;*
                      3 ;* THIS SIMPLE EXAMPLE READS TWO SECTORS FROM THE DISK (TRACK 28H,
                      4 ;* SECTORS 7 AND 8) INTO A BUFFER IN RAM.  IT THEN WRITES THE
                      5 ;* CONTENTS OF THAT BUFFER BACK TO THE DISK AT TRACK 40H, SECTORS
                      6 ;* 18H AND 19H.
                      7 ;*
                      8 ;*************************************************************
                      9 ;
  6000                10          ORG     6000H           ; ARBITRARY ORIGIN POINT
                      11 CLEAR:
  6000 CDD661         12          CALL    DKSTAT          ; GET CURRENT DISK STATUS
  6003 E604           13          ANI     4H              ; OPERATION COMPLETE?
  6005 CA1160         14          JZ      BEGIN           ; BRANCH IF PREVIOUS OPERATION COMPLETE
  6008 CDE261         15          CALL    RTYPE           ; OTHERWISE, CLEAR THE DISK OF
  600B CDDC61         16          CALL    RBYTE           ;    OPERATIONS
  600E C30060         17          JMP     CLEAR
                      18 BEGIN:
  6011 011E60         19          LXI     B,RIOPB         ; LOAD ADDRESS OF RIOPB IN B,C REGISTERS
  6014 CD2C61         20          CALL    ISDDR
  6017 012560         21          LXI     B,WIOPB         ; LOAD ADDRESS OF WIOPB
  601A CD2C61         22          CALL    ISDDR
  601D 76             23          HLT
                      24 ;
                      25 RIOPB:                           ; IOPB FOR READ
  601E 00             26          DB      0               ; IO CONTROL WORD
  601F 04             27          DB      READ            ; READ INSTRUCTION
  6020 02             28          DB      2               ; NUMBER OF SECTORS
  6021 28             29          DB      28H             ; TRACK ADDRESS
  6022 07             30          DB      7               ; SECTOR ADDRESS
  6023 2C60           31          DW      BUFFER          ; BUFFER ADDRESS
                      32 WIOPB:               ; IOPB FOR WRITE
  6025 00             33          DB      0               ; IO CONTROL WORD
  6026 06             34          DB      WRITE           ; WRITE INSTRUCTION
  6027 02             35          DB      2               ; NUMBER OF SECTORS
  6028 40             36          DB      40H             ; TRACK ADDRESS
  6029 18             37          DB      18H             ; SECTOR ADDRESS
  602A 2C60           38          DW      BUFFER          ; BUFFER ADDRESS
  0100                39 BUFFER:  DS      256
                      40
                      41 ;*************************************************************
                      42 ;*
                      43 ;*  PROCEDURE NAME: ISDDR (INTEGRATED SINGLE DENSITY DISK DRIVER)
                      44 ;*  PROCESS:  TRANSMIT THE IOPB, ONE BYTE AT A TIME, TO THE ISD.
                      45 ;*            IF THE INSTRUCTION TO THE DISK IS A DATA TRANSFER
                      46 ;*            (I.E. READ DATA, FORMAT, WRITE DATA, WRITE DELETED
                      47 ;*            DATA) THEN TRANSFER THE DATA, ONE BYTE AT A TIME,
                      48 ;*            TO/FROM THE ISD.
                      49 ;*  INPUT:    B-REG CONTAINS MSB OF IOPB
                      50 ;*            C-REG CONTAINS LSB OF IOPB
                      51 ;*  OUTPUT:   THE IOPB IS TRANSMITTED TO THE ISD.  DATA IS TRANSFERED
                      52 ;*            TO/FROM THE ISD AS REQUIRED.
```

```
ISIS-II 0000/0005 MACRO ASSEMBLER, V2.0        MODULE   PAGE    2

 LOC  OBJ          SEQ          SOURCE STATEMENT

                    53 ;*
                    54 ;*****************************************************************
                    55 ; IOC INTERFACE COMMANDS
 00C1               56 IOCS     EQU     0C1H            ; IOC INPUT DBB STATUS PORT
 00C1               57 IOCC     EQU     0C1H            ; IOC OUTPUT CONTROL COMMAND PORT
 00C0               58 IOCI     EQU     0C0H            ; IOC INPUT DATA (FROM DBB) PORT
 00C0               59 IOCO     EQU     0C0H            ; IOC OUTPUT DATA (TO DBB) PORT
 0004               60 FB       EQU     00000100B       ; FLAG B - SLAVE IS BUSY, MASTER IS LOCKED OUT
 0002               61 IBF      EQU     00000010B       ; SLAVE INPUT BUFFER IS FULL
 0001               62 OBF      EQU     00000001B       ; SLAVE OUTPUT BUFFER IS FULL
 0015               63 WPBC     EQU     15H             ; WRITE PARAMETER BLOCK COMMAND
 0016               64 WPCC     EQU     16H             ; WRITE PARAMETER BLOCK COMMAND CONTINUATION
 0017               65 WDBC     EQU     17H             ; WRITE DATA BLOCK COMMAND
 0018               66 WDCC     EQU     18H             ; RESERVED
 0019               67 RDBC     EQU     19H             ; READ DATA BLOCK COMMAND
 001A               68 RDCC     EQU     1AH             ; RESERVED
 001B               69 RRSTS    EQU     1BH             ; READ RESULT STATUS COMMAND
 001C               70 RDSTS    EQU     1CH             ; READ DEVICE STATUS COMMAND
                    71
                    72 ; PSEUDO INTERRUPT INSTRUCTIONS
 000D               73 DISABL   EQU     0DH             ; DISABLE INTERRUPTS
 0005               74 ENABL    EQU     05H             ; ENABLE INTERRUPTS
 00FF               75 CPUC     EQU     0FFH            ; PORT FOR THESE INSTRUCTIONS
                    76
                    77 ; DISK INSTRUCTIONS
 0001               78 SEEK     EQU     1H              ; SEEK INSTRUCTION
 0002               79 FORMAT   EQU     2H              ; FORMAT INSTRUCTION
 0003               80 RECALB   EQU     3H              ; RECALIBRATE INSTRUCTION
 0004               81 READ     EQU     4H              ; READ DATA INSTRUCTION
 0005               82 VERIFY   EQU     5H              ; VERIFY CRC INSTRUCTION
 0006               83 WRITE    EQU     6H              ; WRITE DATA INSTRUCTION
 0007               84 WRITED   EQU     7H              ; WRITE DELETED DATA INSTRUCTION
                    85
                    86 ; LAYOUT OF THE I/O PARAMETER BLOCK (IOPB)
                    87 ;         ONLY THE FIRST FIVE BYTES OF THE IOPB ARE TRANSMITTED TO THE ISD
                    88 ;         IOCW    BYTE    CHANNEL WORD
                    89 ;         IOINS   BYTE    DISKETTE INSTRUCTION
                    90 ;         NSEC    BYTE    NUMBER OF SECTORS
                    91 ;         TADR    BYTE    TRACK ADDRESS
                    92 ;         SADR    BYTE    SECTOR ADDRESS
                    93 ;         BUF     ADDRESS BUFFER ADDRESS
                    94
                    95
                    96 ISDDRV:
 612C C5            97          PUSH    B               ; SAVE B,C (I.E. ADDRESS OF IOPB)
 612D 03            98          INX     B               ; BC POINTS TO IOINS BYTE
 612E 0A            99          LDAX    B               ; A-REG CONTAINS IOINS
 612F FE04         100          CPI     READ            ; IS IT A READ INSTRUCTION?
 6131 C26561       101          JNZ     ISD1            ; JUMP IF IT ISN'T
 6134 E1           102          POP     H               ; OTHERWISE, PROCESS THE READ COMMAND
                   103                                  ; HL CONTAINS IOPB ADDRESS
 6135 E5           104          PUSH    H               ; SAVE THE ADDRESS OF THE IOPB
 6136 23           105          INX     H
 6137 23           106          INX     H
 6138 56           107          MOV     D,M             ; SAVE THE NUMBER OF SECTORS IN D-REG
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    3


 LOC  OBJ          SEQ          SOURCE STATEMENT

 6139 E1            108          POP     H               ; HL NOW POINTS TO BEGINNING OF THE IOPB
 613A D5            109          PUSH    D               ; SAVE THE NUMBER OF SECTORS
 613B CD8A61        110          CALL    TRIOPB          ; TRANSMIT THE IOPB
 613E 23            111          INX     H               ; HL NOW POINTS TO BUFFER ADDRESS
                    112                                  ;    ENTRY OF THE IOPB
 613F 5E            113          MOV     E,M             ; E CONTAINS LSB OF BUFFER ADDRESS
 6140 23            114          INX     H               ; HL NOW CONTAINS LSB OF BUFFER ADDRESS
 6141 56            115          MOV     D,M             ; D CONTAINS MSB OF BUFFER ADDRESS
 6142 EB            116          XCHG                    ; HL NOW POINTS TO THE BUFFER ITSELF
 6143 0619          117          MVI     B,RDBC          ; LOAD THE READ DATA BLOCK COMMAND
 6145 CD8C62        118          CALL    IOCCOM          ; OUTPUT IT
 6148 D1            119          POP     D               ; D-REG CONTAINS NUMBER OF SECTORS TO BE TRANSFERRED
                    120 RDLP1:
 6149 1E80          121          MVI     E,128           ; SET COUNTER FOR NUMBER OF BYTES PER SECTOR
                    122 RDLP2:
 614B DBC1          123          IN      IOCS            ; INPUT DBB STATUS
 614D E607          124          ANI     FB OR IBF OR OBF; MASK OFF STATUS FLAGS
 614F FE01          125          CPI     OBF             ; TEST FOR SLAVE DONE; SOMETHING FOR THE MASTER
 6151 C24861        126          JNZ     RDLP2           ; LOOP UNTIL SLAVE IS READY
 6154 DBC8          127          IN      IOCI            ; OTHERWISE, INPUT THE DATA FROM THE DBB
 6156 77            128          MOV     M,A             ; STORE THE DATA IN MEMORY AT THE BUFFER LOCATION
 6157 23            129          INX     H               ; HL POINTS TO THE NEXT BYTE IN THE BUFFER
 6158 1D            130          DCR     E               ; DECREMENT THE LENGTH REMAINING IN ONE SECTOR
 6159 C24861        131          JNZ     RDLP2           ; CONTINUE READING UNTIL WHOLE SECTOR READ
 615C 15            132          DCR     D               ; ALL OF A SECTOR HAS BEEN READ. DECREMENT THE
                    133                                  ;    COUNT OF THE TOTAL NUMBER OF SECTORS
                    134                                  ;    REMAINING TO BE READ
 615D C24961        135          JNZ     RDLP1           ; READ THE NEXT SECTOR
 6160 3E85          136          MVI     A,ENABL         ; ENABLE INTERRUPTS
 6162 D3FF          137          OUT     CPUC
 6164 C9            138          RET                     ; RETURN; READ DISK OPERATION COMPLETED
                    139                                  ;-----------------------------------------------------
                    140 ISD1:                            ; WE KNOW IT IS NOT A READ INSTRUCTION
                    141                                  ;    A-REG CONTAINS THE IOINS
 6165 FE02          142          CPI     FORMAT          ; IS IT A FORMAT INSTRUCTION
 6167 CA7461        143          JZ      ISD2            ; JUMP IF IT IS
 616A FE06          144          CPI     WRITE           ; IS IT A WRITE DATA INSTRUCTION?
 616C CA7461        145          JZ      ISD2            ; JUMP IF IT IS
 616F FE07          146          CPI     WRITED          ; IS IT A WRITE DELETED DATA INSTRUCTION?
 6171 C2AD61        147          JNZ     ISD3            ; JUMP IF IT ISN'T (THEREFORE, THE INSTRUCTION IS A
                    148                                  ;    SEEK, RECALB, OR VERIFY)
                    149 ISD2:                            ; A DATA TRANSFER INSTRUCTION INVOLVING A
                    150                                  ;    WRITE TO THE DISKETTE
 6174 E1            151          POP     H               ; HL CONTAINS THE IOPB ADDRESS
 6175 E5            152          PUSH    H               ; SAVE THE IOPB ADDRESS
 6176 23            153          INX     H               ; HL NOW POINTS TO IOINS
 6177 7E            154          MOV     A,M             ; A-REG CONTAINS THE INSTRUCTION
 6178 FE02          155          CPI     FORMAT          ; IS IT A FORMAT INSTRUCTION
 617A 23            156          INX     H               ; HL NOW POINTS TO NSEC
 617B CA8261        157          JZ      ISD2A           ; JUMP IF IT IS A FORMAT INSTRUCTION
 617E 4E            158          MOV     C,M             ; MOVE NUMBER OF SECTORS TO B-REG IF NOT A
                    159                                  ;    FORMAT INSTRUCTION
 617F C38461        160          JMP     ISD2B           ; SKIP FORMAT STUFF
                    161 ISD2A:
 6182 0E01          162          MVI     C,01            ; SINCE IT IS A FORMAT INSTRUCTION, MOVE A 1 TO
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    4

 LOC  OBJ          SEQ          SOURCE STATEMENT

                    163                                ;    THE C-REG (EXPLANATION: FOR TWO BOARD DISK
                    164                                ;    CONTROLLERS, IT IS POSSIBLE TO HAVE AN INVALID
                    165                                ;    VALUE IN THE NUMBER OF SECTORS BYTE OF THE
                    166                                ;    IOPB AND THIS WILL NOT AFFECT THE OPERATION
                    167                                ;    OF THE CONTROLLERS. FOR THE 8271,HOWEVER,
                    168                                ;    WE MUST HAVE A VALID VALUE.)
                    169 ISD2B:
6184 23             170          INX     H             ; HL NOW POINTS TO THE TADR
6185 23             171          INX     H             ; HL NOW POINTS TO SADR
6186 23             172          INX     H             ; HL NOW POINTS TO BUFFER ADDRESS
6187 5E             173          MOV     E,M           ; LOAD THE LSB OF BUFFER ADDRESS INTO E
6188 23             174          INX     H             ; HL NOW POINTS TO MSB OF BUFFER ADDRESS
6189 56             175          MOV     D,M           ; LOAD THE MSB OF BUFFER ADDRESS INTO D
618A EB             176          XCHG                  ; HL NOW POINTS TO FIRST BYTE OF THE BUFFER
618B 0617           177          MVI     B,WDBC        ; LOAD THE WRITE DATA BLOCK COMMAND
618D CD0C62         178          CALL    IOCCOM        ; OUTPUT THE COMMAND
6190 CDB261         179          CALL    IOCRDY        ; MAKE SURE DBB IS CLEAR AND SLAVE IS IDLE
6193 79             180          MOV     A,C           ; LOAD NUMBER OF SECTORS TO BE WRITTEN
6194 D3C0           181          OUT     IOCO          ; OUTPUT DATA TO THE DBB
                    182 WRLP1:
6196 F5             183          PUSH    PSW           ; SAVE THE COUNT OF THE NUMBER OF SECTORS
6197 1680           184          MVI     D,128         ; SET D-REG COUNTER TO NUMBER OF BYTES/SECTOR
                    185 WRLP2:
6199 CDB261         186          CALL    IOCRDY        ; MAKE SURE DBB IS CLEAR AND SLAVE IS IDLE
619C 7E             187          MOV     A,M           ; LOAD DATA TO BE WRITTEN
619D D3C0           188          OUT     IOCO          ; OUTPUT DATA TO THE DBB
619F 23             189          INX     H             ; POINT TO THE NEXT BYTE OF DATA
61A0 15             190          DCR     D             ; ARE WE DONE TRANSFERRING A SINGLE SECTOR?
61A1 C29961         191          JNZ     WRLP2         ; JUMP BACK IF NOT
61A4 F1             192          POP     PSW           ; RESTORE COUNT OF NUMBER OF SECTORS
61A5 3D             193          DCR     A             ; ARE WE DONE TRANSFERRING ALL THE SECTORS?
61A6 C29661         194          JNZ     WRLP1         ; JUMP BACK IF NOT
61A9 3E05           195          MVI     A,ENABL       ; ENABLE INTERRUPTS
61AB D3FF           196          OUT     CPUC
                    197 ISD3:                          ; ANY DISKETTE INSTRUCTION EXCEPT A READ
                    198                                ;    IN ANY CASE, TRANSMIT THE IOPB NOW
61AD E1             199          POP     H             ; HL CONTAINS THE ADDRESS OF THE IOPB
61AE CDBA01         200          CALL    TRIOPB        ; TRANSMIT THE IOPB
61B1 C9             201          RET                   ; RETURN
                    202 ;------
                    203 IOCRDY:
61B2 DBC1           204          IN      IOCS          ; INPUT DBB STATUS
61B4 E607           205          ANI     F0 OR OBF OR IBF; TEST FOR SLAVE PROCESSOR READY
61B6 C2B261         206          JNZ     IOCRDY        ; CONTINUE TO LOOP UNTIL IT IS READY
61B9 C9             207          RET
                    208 ;*****************************************************************
                    209 ;*
                    210 ;* PROCEDURE NAME: TRIOPB (TRANSMIT IOPB TO ISD)
                    211 ;* PROCESS: TRANSMIT THE IOPB TO THE 8271 INTEGRATED SINGLE DENSITY
                    212 ;*           CONTROLLER   THIS PROCEDURE IS CALLED ONLY BY PROCEDURE ISDDR.
                    213 ;* INPUT: HL CONTAINS ADDRESS OF THE IOPB
                    214 ;* OUTPUT: TRANSMIT THE IOPB
                    215 ;*         HL POINTS TO SADR OF IOPB
                    216 ;* MODIFIED: A, FLAGS, B, C, D, HL
                    217 ;*
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    5

  LOC  OBJ          SEQ          SOURCE STATEMENT

                     218 ;*************************************************************************
                     219 TRIOPB:
  61BA 4E            220          MOV     C,M              ; C-REG CONTAINS IOCW
  61BB 0615          221          MVI     B,WPBC           ; LOAD WRITE PARAMETER BLOCK COMMAND
  61BD CDFA61        222          CALL    IOCDR2           ; ISSUE THE COMMAND AND IOCW
  61C0 1604          223          MVI     D,4              ; D-REG CONTAINS COUNTER OF 4
                     224 TRLOOP:
  61C2 23            225          INX     H                ; HL POINTS TO NEXT BYTE IN IOPB
  61C3 4E            226          MOV     C,M              ; C-REG CONTAINS NEXT BYTE IN IOPB
                     227                                   ; WHEN D = 4, C = IOINS
                     228                                   ;      D = 3, C = NSEC
                     229                                   ;      D = 2, C = TADR
                     230                                   ;      D = 1, C = SADR
  61C4 0616          231          MVI     B,WPCC           ; LOAD CONTINUATION COMMAND
  61C6 CDFA61        232          CALL    IOCDR2           ; ISSUE THE COMMAND AND THE DATA
  61C9 15            233          DCR     D
  61CA C2C261        234          JNZ     TRLOOP           ; JUMP IF WE HAVEN'T FINISHED
                     235 TRWAIT:                           ; WAIT FOR RETURN OF STATUS BYTE
  61CD CDD661        236          CALL    DKSTAT           ; SEE IF THE BIT INDICATING OPERATION
                     237                                   ;    COMPLETE IS TURNED ON
  61D0 E604          238          ANI     4H
  61D2 CACD61        239          JZ      TRWAIT           ; KEEP LOOPING UNTIL OPERATION COMPLETED
  61D5 C9            240          RET                      ; RETURN OTHERWISE
                     241 ;*************************************************************************
                     242 ;*
                     243 ;* PROCEDURE NAME: DK$STAT (DISK STATUS)
                     244 ;* PROCESS: RETURN THE DISK DEVICE STATUS
                     245 ;* INPUT:
                     246 ;* OUTPUT: A-REG CONTAINS THE STATUS BYTE
                     247 ;*
                     248 ;*************************************************************************
                     249 DKSTAT:
  61D6 061C          250          MVI     B,RDSTS          ; LOAD THE RDSTS COMMAND
  61D8 CDE561        251          CALL    IOCDR1
  61DB C9            252          RET                      ; RETURN WITH THE ISD STATUS BYTE
                     253 ;*************************************************************************
                     254 ;*
                     255 ;* PROCEDURE NAME: R$BYTE
                     256 ;* PROCESS: RETURN WITH THE RESULT BYTE OF A DISK I/O OPERATION
                     257 ;* INPUT:
                     258 ;* OUTPUT: A-REG CONTAINS THE RESULT BYTE
                     259 ;*
                     260 ;*************************************************************************
                     261 RBYTE:
  61DC 061B          262          MVI     B,RRSTS          ; LOAD THE READ RESULT STATUS COMMAND
  61DE CDE561        263          CALL    IOCDR1
  61E1 C9            264          RET                      ; RETURN WITH THE ISD RESULT STATUS BYTE
                     265 ;*************************************************************************
                     266 ;*
                     267 ;* PROCEDURE NAME: R$TYPE
                     268 ;* PROCESS: RETURN THE RESULT TYPE OF A DISK OPERATION
                     269 ;* INPUT:
                     270 ;* OUTPUT: A-REG CONTAINS THE RESULT TYPE
                     271 ;*
                     272 ;*************************************************************************
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    6


 LOC  OBJ          SEQ          SOURCE STATEMENT

                    273 RTYPE:
 61E2 3E00          274          MVI     A,0              ; RETURN A ZERO SINCE ISD DOES NOT REALLY HAVE A RESULT TYPE
 61E4 C9            275          RET
                    276 ;*****************************************************************************
                    277 ;*
                    278 ;* PROCEDURE NAME: IOCDR1
                    279 ;* PROCESS: GET DEVICE STATUS OR DATA FROM DISK
                    280 ;* INPUT: B CONTAINS THE IOC COMMAND (STATUS REQUES OR INPUT
                    281 ;*        DATA REQUEST
                    282 ;* OUTPUT: A-REG CONTAINS THE REQUESTED INFORMATION
                    283 ;* MODIFIED: A,FLAGS,B
                    284 ;*
                    285 ;*****************************************************************************
                    286 IOCDR1:
 61E5 CD0C62        287          CALL    IOCCOM           ; OUTPUT "GET DEVICE STATUS COMMAND" OR
                    288                                   ;    "INPUT DATA COMMAND" TO IOC CONTROL PORT
                    289 IOCXXX:
 61E8 DBC1          290          IN      IOCS             ; INPUT DBB STATUS
 61EA E607          291          ANI     IBF OR OBF OR F0 ; MASK OFF STATUS FLAGS
 61EC FE01          292          CPI     OBF              ; TEST FOR SLAVE DONE: SOMETHING FOR THE MASTER
 61EE C2E861        293          JNZ     IOCXXX           ; IF NOT, CONTINUE TO LOOP
 61F1 DBC0          294          IN      IOCI             ; OTHERWISE, INPUT THE DATA FROM THE DBB
 61F3 F5            295          PUSH    PSW              ; SAVE A-REG
 61F4 3E05          296          MVI     A,ENABL          ; ENABLE INTERRUPTS
 61F6 D3FF          297          OUT     CPUC
 61F8 F1            298          POP     PSW              ; RESTORE A-REG
 61F9 C9            299          RET
                    300 ;*****************************************************************************
                    301 ;*
                    302 ;* PROCEDURE NAME: IOCDR2
                    303 ;* PROCESS: OUTPUT DATA TO THE DISK
                    304 ;* INPUT: B CONTAINS THE COMMAND TO OUTPUT THE DATA
                    305 ;*        C CONTAINS THE DATA TO BE OUTPUT
                    306 ;* OUTPUT:
                    307 ;* MODIFIED: A,FLAGS,B,C
                    308 ;*
                    309 ;*****************************************************************************
                    310 IOCDR2:
 61FA CD0C62        311          CALL    IOCCOM           ; OUTPUT "OUTPUT DATA COMMAND" TO IOC
                    312                                   ;    CONTROL PORT
                    313 IOCYYY:
 61FD DBC1          314          IN      IOCS             ; INPUT DBB STATUS
 61FF E607          315          ANI     IBF OR F0 OR OBF ; TEST FOR SLAVE PROCESSOR READY
 6201 C2FD61        316          JNZ     IOCYYY           ; CONTINUE TO LOOP UNTIL IT IS READY
 6204 79            317          MOV     A,C              ; LOAD DATA TO BE WRITTEN
 6205 D3C0          318          OUT     IOCO             ; OUTPUT DATA TO THE DBB
 6207 3E05          319          MVI     A,ENABL          ; ENABLE INTERRUPTS
 6209 D3FF          320          OUT     CPUC
 620B C9            321          RET
                    322 ;*****************************************************************************
                    323 ;*
                    324 ;* PROCEDURE NAME: IOCCOM
                    325 ;* PROCESS: OUTPUT COMMAND TO THE IOC
                    326 ;* INPUT: B CONTAINS THE COMMAND
                    327 ;* OUTPUT:
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    7

  LOC  OBJ          SEQ          SOURCE STATEMENT

                     328 ;* MODIFIED: A,FLAGS
                     329 ;*
                     330 ;*****************************************************************
                     331 IOCCOM:
  620C 3E0D          332          MVI     A,DISABL          ; BLOCK ALL INTERRUPTS
  620E D3FF          333          OUT     CPUC
                     334 IOCZZZ:
  6210 DBC1          335          IN      IOCS              ; INPUT DBB STATUS
  6212 E607          336          ANI     F0 OR IBF OR OBF; TEST FOR SLAVE PROCESSOR IDLE
  6214 C21062        337          JNZ     IOCZZZ            ; LOOP UNTIL IT IS IDLE
  6217 78            338          MOV     A,B               ; LOAD THE COMMAND
  6218 D3C1          339          OUT     IOCC              ; OUTPUT COMMAND TO IOC CONTROL PORT
  621A C9            340          RET
                     341 ;*****************************************************************
  6000               342          END     CLEAR

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS
BEGIN   A 6011    BUFFER A 602C    CLEAR   A 6000    CPUC    A 00FF    DISABL A 000D    DKSTAT A 61D6    ENABL   A 0005
F0      A 0004    FORMAT A 0002    IBF     A 0002    IOCC    A 00C1    IOCCOM A 620C    IOCDR1 A 61E5    IOCDR2 A 61FA
IOCI    A 00C0    IOCO   A 00C0    IOCRDY A 6102    IOCS    A 00C1    IOCXXX A 61E8    IOCYYY A 61FD    IOCZZZ A 6210
ISD1    A 6165    ISD2   A 6174    ISD2A  A 6182    ISD2B   A 6184    ISD3   A 61AD    ISDDR  A 612C    OBF     A 0001
RBYTE   A 61DC    RDBC   A 0019    RDCC   A 001A    RDLP1   A 6149    RDLP2  A 614B    RDSTS  A 001C    READ    A 0004
RECALB  A 0003    RIOPB  A 601E    RRSTS  A 0018    RTYPE   A 61E2    SEEK   A 0001    TRIOPB A 610A    TRLOOP  A 61C2
TRWAIT  A 61CD    VERIFY A 0005    WDBC   A 0017    WDCC    A 0018    WIOPB  A 6025    WPBC   A 0015    WPCC    A 0016
WRITE   A 0006    WRITED A 0007    WRLP1  A 6196    WRLP2   A 6199

ASSEMBLY COMPLETE,  NO ERRORS
```

APPENDIX E
CONNECTOR PIN ASSIGNMENTS

This appendix identifies the pin assignments for all user-interface connectors for the Intellec Series II Microcomputer Development System and defines the dc drive and load characteristics for the individual signals. The Multibus interface is internal to the development system and is available on connectors J2 through J6 of the card cage. Table E-1 identifies the Multibus interface pin assignments and tables E-2 and E3 define the Multibus interface dc signal characteristics for the IPB and IPC, respectively. Tables E-4 through E-9 define the pin assignments and dc characteristics for the rear panel peripheral connectors (J2 through J7).

Table E-1. MULTIBUS® Interface Pin Assignments

Board Component Side			Board Circuit Side		
Pin	**Mnemonic**	**Description**	**Pin**	**Mnemonic**	**Description**
1	GND	Signal ground	**2**	GND	Signal ground
3	+5	+5 VDC	**4**	+5	+5 VDC
5	+5	+5 VDC	**6**	+5	+5 VDC
7	+12	+12 VDC	**8**	+12	+12 VDC
9	−5	−5 VDC	**10**	−5	−5 VDC
11	GND	Signal ground	**12**	GND	Signal ground
13	BCLK/	Bus Clock	**14**	INIT/	Initialize
15	BPRN/	Bus Priority In	**16**	BPRO/*	Bus Priority Out
17	BUSY/	Bus Busy	**18**	BREQ/	Bus Request
19	MRDC/	Memory Read Command	20	MWTC/	Memory Write Command
21	IORC/	I/O Read Command	**22**	IOWC/	I/O Write Command
23	XACK/	Transfer Acknowledge	24	INH1/	Inhibit (disable) RAM
25	AACK/	Advanced Acknowledge	26	INH2/	Inhibit (disable) ROM
27	BHEN/	Byte High Enable	**28**	ADR10/	Address Extension Lines
29	CBRQ/**	Common Bus Request	30	ADR11/	
31	CCLK/	Constant Clock	**32**	ADR12/	
33	INTA/*	Interrupt Acknowledge	34	ADR13/	
35	INT6/	Interrupt Requests	**36**	INT7/	Interrupt Requests
37	INT4/		**38**	INT5/	
39	INT2/		**40**	INT3/	
41	INT0/		**42**	INT1/	
43	ADRE/	Address Lines	**44**	ADRF/	Address Lines
45	ADRC/		**46**	ADRD/	
47	ADRA/		**48**	ADRB/	
49	ADR8/		**50**	ADR9/	
51	ADR6/		**52**	ADR7/	
53	ADR4/		**54**	ADR5/	
55	ADR2/		**56**	ADR3/	
57	ADR0/		**58**	ADR1/	
59	DATE/	Data Lines	**60**	DATF/	Data Lines
61	DATC/		**62**	DATD/	
63	DATA/		**64**	DATB/	
65	DAT8/		**66**	DAT9/	
67	DAT6/		**68**	DAT7/	
69	DAT4/		**70**	DAT5/	
71	DAT2/		**72**	DAT3/	
73	DAT0/		**74**	DAT1/	
75	GND	Signal ground	**76**	GND	Signal ground
77	−10	−10 VDC	**78**	−10	−10 VDC
79	−12	−12 VDC	**80**	−12	−12 VDC
81	+5	+5 VDC	**82**	+5	+5 VDC
83	+5	+5 VDC	**84**	+5	+5 VDC
85	GND	Signal ground	**86**	GND	Signal ground

* Not implemented on IPB/IPC

** Only implemented on IPC

Table E-2. IPB Signal DC Characteristics

Signal Mnemonic	Current Drive		Current Load		Type	Termination
	Low (I_{OL})	High (I_{OH})	Low (I_{IL})	High (I_{IH})		
AACK/	20mA	N/A	−4mA	100μA	Open Collector	370Ω pullup
ADR0/-ADR7/	24mA	−15mA	−0.5mA	70μA	Three-State	2.2kΩ pullup
ADR8/-ADRF/	24mA	−15mA	−0.45m A	30μA	Three-State	2.2kΩ pullup
ADR10/-ADR13/	N/A	N/A	N/A	N/A	N/A	2.2kΩ pullup
BCLK/, CCLK/	60mA	−3mA	−0.5mA	100μA	TTL	220/330Ω on backplane*
BHEN/	N/A	N/A	N/A	N/A	N/A	2.2kΩ pullup
BPRN1/-BPRN9/	20mA	−1mA	0	0	TTL	None
BREQ1/-BREQ9/	N/A	N/A	−3.2mA	80μA	N/A	1kΩ pullup
BUSY/	20mA	N/A	0	0	Open Collector	1kΩ pullup
DAT0/-DAT7/	25mA	−10mA	−0.5mA	80μA	Three-State	2.2kΩ pullup
DAT8/-DATF/	N/A	N/A	N/A	N/A	N/A	2.2kΩ pullup
INH1/	15mA	N/A	−1.6mA	40μA	TTL	1kΩ pullup
INH2/	15mA	N/A	0	0	TTL	1kΩ pullup
INIT/	60mA	−3mA	0	0	TTL	None
INT0/-INT7/	40mA	N/A	−0.2mA	20μA	Open Collector	1kΩ pullup
IORC/, IOWC/	32mA	−2mA	−0.45mA	60μA	Three-State	1kΩ pullup
MRDC/, MWTC/	32mA	−2mA	−1.6mA	40μA	Three-State	1kΩ pullup
XACK/	20mA	N/A	−2mA	50μA	Open Collector	510Ω pullup

*A 150Ω/100pF series RC termination network is also installed on the backplane.

Table E-3. IPC Signal DC Characteristics

Signal Mnemonic	Current Drive		Current Load		Type	Termination
	Low (I_{OL})	High (I_{OH})	Low (I_{IL})	High (I_{IH})		
AACK/	16mA	−5.2mA	−0.4mA	20μA	Three-State	510Ω pullup
ADR0/-ADRF/	32mA	−5mA	−0.4mA	50μA	Three-State	2.2kΩ pullup
ADR10/-ADR13/	N/A	N/A	−0.4mA	20μA	N/A	2.2kΩ pullup
BCLK/, CCLK/	60mA	−3mA	−0.5mA	100μA	TTL	220/330Ω on backplane*
BHEN/	N/A	N/A	−0.4mA	20μA	N/A	2.2kΩ pullup
BPRN1/-BPRN9/	20mA	−1mA	0	0	TTL	None
BREQ1/-BREQ9/	N/A	N/A	−3.2mA	80μA	N/A	1kΩ pullup
BUSY/	20mA	N/A	0	0	Open Collector	1kΩ pullup
CBRQ/	20mA	N/A	0	0	Open Collector	1kΩ pullup
DAT0/-DAT7/	20mA	−5mA	−0.95mA	230μA	Three-State	2.2kΩ pullup
DAT8/-DATF/	20mA	−5mA	−0.25mA	70μA	Three-State	2.2kΩ pullup
INH1/	48mA	N/A	−0.4mA	0	Open Collector	1kΩ pullup
INH2/	48mA	N/A	0	N/A	Open Collector	1kΩ pullup
INIT/	48mA	N/A	0	N/A	Open Collector	2.2kΩ pullup
INT0/-INT7/	40mA	N/A	−0.2mA	20μA	Open Collector	1kΩ pullup
IORC/, IOWC/	32mA	−2mA	−0.2mA	20μA	Three-State	1kΩ pullup
MRDC/, MWTC/	32mA	−2mA	−2mA	50μA	Three-State	1kΩ pullup
XACK/	16mA	−5.2mA	−0.4mA	20μA	Three-State	510Ω pullup

*A 150Ω/100pF series RC termination network is also installed on the backplane.

Table E-4. SERIAL CH1/TTY Pin Assignments (Connector J2)

Pin	Signal	Function	Current Drive		Current Load	
			Low (I_{OL})	High (I_{OH})	Low (I_{IL})	High (I_{IH})
1	PROT GND	Protective Ground				
2	RxD (RS232)	Transmitted Data In			−4mA*	4mA*
3	TxD (RS232)	Received Data Out	6mA	−6mA		
4	RTS (RS232)	Request to Send	6mA	−6mA		
5	CTS (RS232)	Clear to Send			−4mA*	4mA*
6	DSR (RS232)	Data Set Ready			−4mA*	4mA*
7	GND	Signal Ground				
8		Not Used				
9		Not Used				
10		Not Used				
11		Not Used				
12	RxD (CURRENT LOOP)	Transmitted Data In			<100μA (ON)	>16mA (OFF)
13	TxD (CURRENT LOOP)	Received Data Out	>22mA (ON)	<100μA (OFF)		
14	TTY RDY	Same as DSR (pin 6)				
15	TxC	Transmit Clock				
16	DTR (CURRENT LOOP)	Data Terminal Ready (Reader Control)	<100μA (ON)	>22mA (OFF)		
17	RxC	Receive Clock			−4mA*	4mA*
18		Not Used				
19		Not Used				
20	DTR (RS232)	Data Terminal Ready	6mA	−6mA		
21	DTR RET	Reader control Return (to −12V)				
22		Not Used				
23		Not Used				
24	RxD RET (CURRENT LOOP)	RxD Return (to + 12V)				
25	TxD RET (CURRENT LOOP)	TxD Return (to −12V)				

Note: The required mating connector is a Cannon DEC-25P (or equivalent).
*At 12.0 volts

Table E-5. SERIAL CH2 Pin Assignments (Connector J3)

Pin	Signal	Function	Current Drive		Current Load	
			Low (I_{OL})	High (I_{OH})	Low (I_{IL})	High (I_{IH})
1	PROT GND	Protective Ground				
2	TxD	Transmitted Data Out	6mA	−6mA		
3	RxD	Received Data In			−1.7mA	1.7mA
4	RTS	Request to Send	6mA	−6mA		
5	CTS	Clear to Send			−1.7mA	1.7mA
6	DSR	Data Set Ready			−1.7mA	1.7mA
7	GND	Signal Ground				
8		Unassigned				
9		Not Used				
10		Not Used				
11	+12V	+12V (requires jumper connection)				
12		Not Used				
13		Not Used				
14		Not Used				
15	TxC	Transmit Clock			−1.7mA	1.7mA
16		Not Used				
17	RxC	Receive Clock			−1.7mA	1.7mA
18		Not Used				
19		Not Used				
20	DTR	Data Terminal Ready	6mA	−6mA		
21		Not Used				
22		Not Used				
23	−12V	−12V (requires jumper connection)				
24	EXT TxC	External Transmit Clock	6mA	−6mA		
25	+5V	+5V (requires jumper connection				

Note: The required mating connector is a Cannon DEC-25P (or equivalent).

Table E-6. PT PUNCH Pin Assignments (Connector J4)

Pin	Signal	Function	Current Drive		Current Load		Termination
			Low (I_{OL})	High (I_{OH})	Low (I_{IL})	High (I_{IH})	
1	DATA TRACK 1/	Output Data Bit 1	15mA	−1mA			
2	DATA TRACK 2/	Output Data Bit 2	15mA	−1mA			
3	DATA TRACK 3/	Output Data Bit 3	15mA	−1mA			
4	DATA TRACK 4/	Output Data Bit 4	15mA	−1mA			
5	DATA TRACK 5/	Output Data Bit 5	15mA	−1mA			
6	DATA TRACK 6/	Output Data Bit 6	15mA	−1mA			
7	DATA TRACK 7/	Output Data Bit 7	15mA	−1mA			
8	DATA TRACK 8/	Output Data Bit 8	15mA	−1mA			
9		Not Used					
10	DIRECTION	Direction Control					1kΩ pullup
11	PUNC COMMAND/	Punch Command	16mA	−800μA			
12	PUNCH READY/	Punch Ready			−10.85μA	−7.4μA	470Ω pullup
13	SYSTEM READY/	System Ready			−10.85μA	−7.4μA	470Ω pullup
14	INPUT MODE SELECT	Select Input Mode					Ground
15	OUTPUT MODE SELECT	Select Output Mode					Ground
16	CHASSIS GND	Chassis Ground					
17	CHASSIS GND	Chassis Ground					
18	GND	Ground					
19		Not Used					
20		Not Used					
21		Not Used					
22		Not Used					
23	GND	Ground					
24		Not Used					
25	GND	Ground					

NOTE: *Slash (/) after signal mnemonic denotes that signal is true when ≤0.4V.
The required mating connector is a Cannon DEC-25P (or equivalent).
Current values include pullup load.

Table E-7. PT READER Pin Assignments (Connector J5)

Pin	Signal	Function	Current Drive		Current Load		Termination
			Low (I_{OL})	High (I_{OH})	Low (I_{IL})	High (I_{IH})	
1	DATA TRACK 1/	Input Data Bit 1			−0.75mA	−0.24mA	10KΩ pullup
2	DATA TRACK 2/	Input Data Bit 2			−0.75mA	−0.24mA	10KΩ pullup
3	DATA TRACK 3/	Input Data Bit 3			−0.75mA	−0.24mA	10KΩ pullup
4	DATA TRACK 4/	Input Data Bit 4			−0.75mA	−0.24mA	10KΩ pullup
5	DATA TRACK 5/	Input Data Bit 5			−0.75mA	−0.24mA	10KΩ pullup
6	DATA TRACK 6/	Input Data Bit 6			−0.75mA	−0.24mA	10KΩ pullup
7	DATA TRACK 7/	Input Data Bit 7			−0.75mA	−0.24mA	10KΩ pullup
8	DATA TRACK 8/	Input Data Bit 8			−0.75mA	−0.24mA	10KΩ pullup
9	DATA READY/	Data Ready			−10.85mA	−7.4mA	470Ω pullup
10		Not Used					
11	GND	Ground					
12	GND	Ground					
13	GND	Ground					
14	SYSTEM READY/	System Ready			−10.85mA	−7.4mA	470Ω pullup
15		Not Used					
16	DR/	Paper Tape Drive Right	16mA	−800μA			
17	DL/	Paper Tape Drive Left	16mA	−800μA			
18		Not Used					
19		Not Used					
20		Not Used					
21		Not Used					
22		Not Used					
23		Not Used					
24	GND	Ground					
25	CHASSIS GND	Chassis Ground					

NOTE: *Slash (/) after signal mnemonic denotes that signal is true when ≤0.4V.
The required mating connector is a Cannon DEC-25P (or equivalent).
Current values include pullup load.

Table E-8. LINE PRINTER Pin Assignments (Connector J6)

Pin	Signal	Function	Current Drive		Current Load		Termination
			Low (I_{OL})	High (I_{OH})	Low (I_{IL})	High (I_{IH})	
1	DATA 1	Output Data Bit 1	15mA	−1mA			
2	DATA 2	Output Data Bit 2	15mA	−1mA			
3	DATA 3	Output Data Bit 3	15mA	−1mA			
4	DATA 4	Output Data Bit 4	15mA	−1mA			
5	DATA 5	Output Data Bit 5	15mA	−1mA			
6	DATA 6	Output Data Bit 6	15mA	−1mA			
7	DATA 7	Output Data Bit 7	15mA	−1mA			
8	DATA 8	Output Data Bit 8	15mA	−1mA			
9	GND	Ground					
10	GND	Ground					
11	GND	Ground					
12	GND	Ground					
13		Not Used					
14	LPT DATA STROBE/	Data Strobe	16mA	−800μA			
15	GND	Ground					
16	ACKNOWLEDGE/				−10.85mA	−7.4mA	470Ω pullup
17	BUSY/	Printer Busy			−10.85mA	−7.4mA	470Ω pullup
18		Not Used					
19	LPT CTL 1/	Control Line 1	16mA	−800μA			
20	LPT CTL 2/	Control Line 2	16mA	−800μA			
21		Not Used					
22	SELECT/	Printer Select			−10.85mA	−7.4mA	470Ω pullup
23		Not Used					
24		Not Used					
25	CHASSIS GND	Chassis Ground					

NOTE: *Slash (/) after signal mnemonic denotes that signal is true when ≤0.4V.
The required mating connector is a Cannon DEC-25P (or equivalent).
Current values include pullup load.

Table E-9. UPP Pin Assignments (Connector J7)

Pin	Signal	Function	Current Drive		Current Load		Termination
			Low (I_{OL})	High (I_{OH})	Low (I_{IL})	High (I_{IH})	
1	GND	Ground					
2	PPACK/	PROM Programmer Acknowledge			−10.6mA	−5.9mA	470KΩ pullup
3	PPRC1/	PROM Programmer Read Control Line 1	16mA	−800μA			
4	PPRC0/	PROM Programmer Read Control Line 2	16mA	−800μA			
5	PRD7/	PROM Read Data Bit 7			−0.75mA	−0.24mA	10KΩ pullup
6	PRD6/	PROM Read Data Bit 6			−0.75mA	−0.24mA	10KΩ pullup
7	PRD5/	PROM Read Data Bit 5			−0.75mA	−0.24mA	10KΩ pullup
8	PRD4/	PROM Read Data Bit 4			−0.75mA	−0.24mA	10KΩ pullup
9	PRD3/	PROM Read Data Bit 3			−0.75mA	−0.24mA	10KΩ pullup
10	PRD2/	PROM Read Data Bit 2			−0.75mA	−0.24mA	10KΩ pullup
11	PRD1	PROM Read Data Bit 1			−0.75mA	−0.24mA	10KΩ pullup
12	PRD0/	PROM Read Data Bit 0			−0.75mA	−0.24mA	10KΩ pullup
13	GND	Ground					
14	INIT/	Initialize	16mA	−800μA			
15	PWD7/	PROM Write Data Bit 7	15mA	−1mA			
16	PWD6/	PROM Write Data Bit 6	15mA	−1mA			
17	PWD5/	PROM Write Data Bit 5	15mA	−1mA			
18	PWD4/	PROM Write Data Bit 4	15mA	−1mA			
19	PWD3/	PROM Write Data Bit 3	15mA	−1mA			
20	PWD2/	PROM Write Data Bit 2	15mA	−1mA			
21	PWD1/	PROM Write Data Bit 1	15mA	−1mA			
22	PWD0/	PROM Write Data Bit 0	15mA	−1mA			
23	PPWC2/	PROM Programmer Write Control Line 2	16mA	−800μA			
24	PPWC1/	PROM Programmer Write Control Line 1	16mA	−800μA			
25	PPWC0/	PROM Programmer Write Control Line 0	16mA	−800μA			

NOTE: *Slash (/) after signal mnemonic denotes that signal is true when ≤0.4V.
The required mating connector is a Cannon DEC-25P (or equivalent).
Current values include pullup load.