IBM

Personal Computer Hardware Reference Library

IBM RT PC Hardware Technical Reference

Volume I

75X0232

Personal Computer
Hardware Reference
Library

IBM RT PC Hardware Technical Reference

Volume I

About This Book

Purpose

This manual describes the IBM RT Personal Computer[1] Product Family Hardware, the various units and components of the product family and how they interact. It provides information about the family system architecture, and about hardware and programming interfaces.

Audience

The information in this manual is for reference. It is intended for hardware and program designers, programmers, engineers, and anyone else who needs to understand the design and operation of the IBM RT[1] PC Product Family.

How to Use This Book

This manual is composed of ten sections. Section 1, System Introduction, introduces major system components in the order they are discussed in subsequent sections. It provides an initial description and understanding of the functions of key components of the product family workstations, and how the workstations are physically configured with various components.

An overview of the system architecture is included with a system block diagram which shows major partitions of the system unit elements such as the system board, processor board, memory boards, Floating-Point Accelerator board. Major system characteristics are described such as a 32-bit system unit, demand paged virtual memory system, high bandwidth interleaved memory and standard I/O.

Section 2 describes the major functions of the processor board. Section 3 provides a functional description of the system memory boards. Section 4 describes the Floating-Point Accelerator board. Section 5 contains information about the system boards, with a description of the differences between each of the boards as used with its unique system unit. Section 6 provides a complete architectural description of the I/O channel. Section 7 describes the functions provided by the system IPL ROM, including a list of the various error codes displayed by diagnostic routines. Section 8 contains information about adapters from the IBM Personal Computer Family, and their compatibility or noncompatibility with the RT PC[1] family. Section 9 provides

---

[1] Trademark of International Business Machines Corporation

Second Edition (September 1986)

Changes are made periodically to the information herein; these changes will be incorporated in new editions of this publication.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

International Business Machines Corporation provides this manual "as is," without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this manual at any time.

Products are not stocked at the address given below. Requests for copies of this product and for technical information about the system should be made to your authorized IBM RT PC dealer.

A reader's comment form is provided at the back of this publication. If the form has been removed, address comments to IBM Corporation, Department 997, 11400 Burnet Road, Austin, Texas 78758. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

information about the locator, keyboard, characters, keystrokes. Section 10 describes the system unit physical dimensions, power, environmental and operating requirements. Section 11 includes detailed specification information for the system processor and memory management unit components.

Appendix A lists the instruction set by mnemonic and provides information for instruction execution time.

Appendix B describes the Advanced Processor Board and how it differs from the original processor board.

A glossary of terms and a bibliography of related publications is included.

An abbreviated index is included. A more complete Section Table of Contents is provided at the beginning of each section.

Related Information

IBM PC Network Technical Reference Manual

IBM RT PC Personal Computer AT Coprocessor Services Technical Reference

IBM RT PC Virtual Resource Manager Technical Reference

IBM Token-Ring Network RT PC Adapter Technical Reference Manual

3278/79 Emulation Adapter Technical Reference Manual

Ordering Additional Copies of This Book

To order additional copies of this publication, use either of the following sources:

- To order from your IBM representative, use Order Number SV21-8024.
- To order from your IBM dealer, use Part Number 75X1072.

A binder is included with the order.

Contents

Figures

Section 1. System Introduction

CONTENTS

About this Section

This section contains an introduction to the IBM RT PC System. The differences between the IBM 6150 System Unit and the IBM 6151 System Unit are described. The components that reside on the system board are discussed along with a description of how they transfer data throughout the system.

General System Description

The IBM RT PC Product Family is a family of workstations supporting both stand-alone and clustered configurations. Each workstation offers significant computational power, coupled to all points addressable displays. The workstations have a common family appearance, use the same software and user interface. A wide range of options and adapters are offered for the workstations, such as data communications, color displays, quality printers, graphic printers, and color plotters. The IBM 6150 System Unit is a small floor standing console and an ergonomically packaged display monitor. The second system, the IBM 6151 System Unit, is a table-top workstation. Each has growth capabilities.

IBM 6150 System Unit

The IBM 6150 System Unit is a 32-bit system unit packaged in a small floor standing console. A IBM 6150 can be configured by selecting one of the following displays and its appropriate adapter board:

- IBM 5151 Personal Computer Display
- IBM 6153 Advanced Monochrome Graphics Display
- IBM 5154 Enhanced Color Display
- IBM 6154 Advanced Color Graphics Display
- IBM 6155 Extended Monochrome Graphics Display.

The system unit console contains the system board, power supply, fixed-disk drives, and diskette drives. The system board has unique pluggable slots for the processor board, the Floating-Point Accelerator board, and for two RT PC memory boards with addressing for up to 16M-bytes of system memory. In addition, eight slots are provided for optional I/O adapters. Two of these eight slots only accommodate adapters from the IBM PC-1 and the PC-XT. The remaining six slots accommodate RT PC adapters, PC-1 and PC-XT adapters, and PC-AT adapters identified in this manual.

A special Coprocessor board, Intel 80286 based, is available as an option for the IBM 6150. This board allows most existing PC applications to run on RT PC. The IBM Personal Computer AT 512 KB Memory Expansion Option can be used with this board.

The system unit accommodates up to five 5.25-inch storage device drives. Three full-high positions are provided, with two positions configured to accept either one or two 5.25-inch half-high diskette drives. The other three positions accommodate a maximum of three fixed-disk drives.

IBM 6151 System Unit

The IBM 6151 System Unit is a 32-bit system unit packaged in a console designed for table top operation. A IBM 6151 can be configured by selecting one of the following displays and its appropriate adapter:

- IBM 5151 Personal Computer Display
- IBM 6153 Advanced Monochrome Graphics Display
- IBM 5154 Enhanced Color Display
- IBM 6154 Advanced Color Graphics Display
- IBM 6155 Extended Monochrome Graphics Display.

The system unit console contains the system board, power supply, fixed-disk drive and diskette drive. The system board has unique pluggable slots for the processor board, Floating-Point Accelerator, and two RT PC memory boards, with addressing for up to of 16M-bytes of system memory. In addition, six slots are provided for optional I/O adapters. One of these slots only accommodates adapters from the IBM PC and XT computers, while the remaining five accommodate RT PC adapters, PC-1 and PC-XT adapters, and IBM PC-AT adapters identified in this manual.

A special Coprocessor board, Intel 80286 based, is available as an option for the IBM 6151. This board allows most existing PC applications to run on RT PC. The IBM Personal Computer AT 512 KB Memory Expansion Option can be used with this board.

The IBM 6151 console accommodates one full high 5.25-inch fixed-disk drive and one half-high 5.25-inch diskette drive.

System Overview

The IBM 6150 System Unit and the IBM 6151 System Unit are virtual memory 32-bit workstations. All units use the same processor board, system memory boards, optional Floating-Point Accelerator, and I/O adapters. A unique system board is used in each unit, and differs only in the number of I/O slots provided and in the built-in RS232C serial port capability.

The following section provides an overview of the RT PC system architecture. This section contains information for the IBM 6151 and and the IBM 6150. Features unique only to the IBM 6150 are also described.

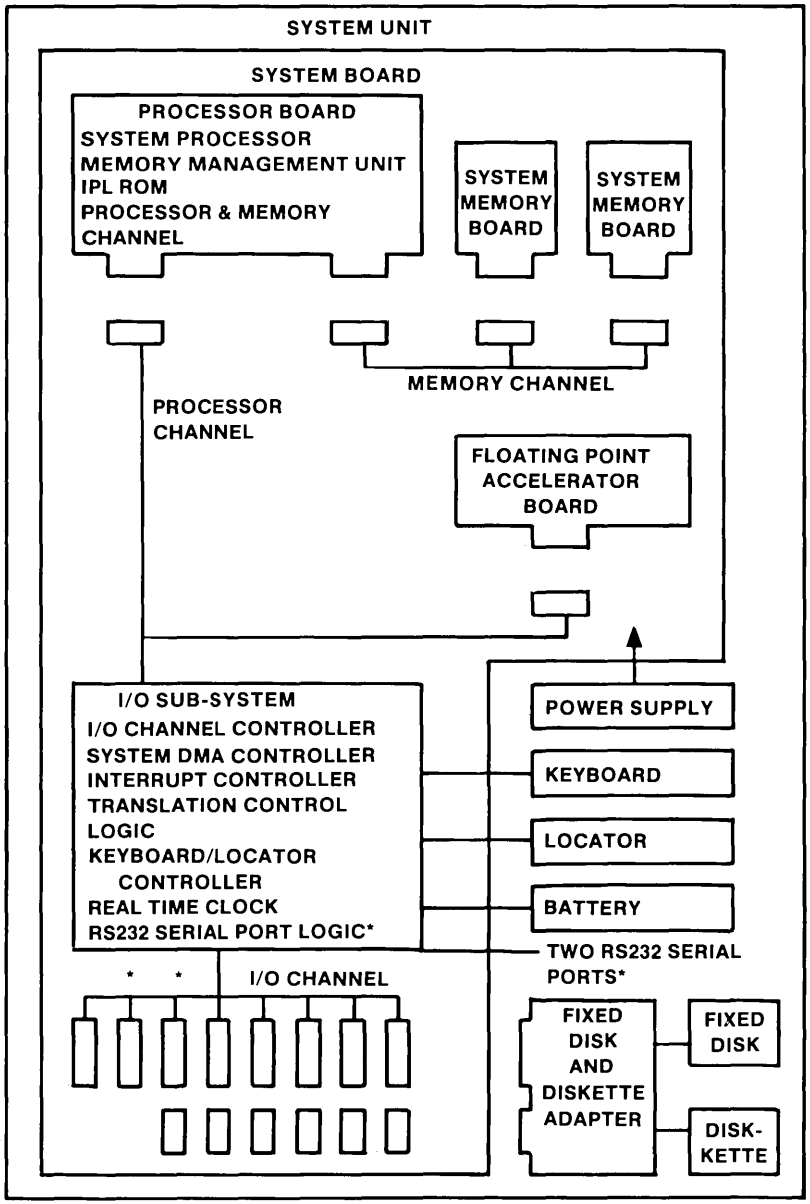


Figure 1-1. System Board Block Diagram

Note: * = IBM 6150 only

System Board

The RT PC system board contains the I/O subsystem, I/O channel, and connectors for the processor board, system memory boards, and optional Floating-Point Accelerator. Unique system boards are used for the IBM 6151 and the IBM 6150, and differ only in the number of I/O slots provided and in the built-in RS232C serial port capability. The same processor board, system memory boards, Floating-Point Accelerator, and I/O adapters are used in both units.

Processor Board

The processor board plugs into the system board and contains the following major components:

- A 32-bit microprocessor
- A memory management unit
- A 64K-byte read-only memory (ROM) for initial program load (IPL)
- Reference and change bits for virtual memory management
- Memory channel and processor channel control and interface logic.

The RT PC processor board includes a system processor and a memory management unit (MMU). Both components use Very Large Scale Integration (VLSI) technology. These two components form a 32-bit virtual memory processor unit, providing advanced memory management functions. The system processor provides sixteen 32-bit general purpose registers and 118 instructions. All internal address and data paths are 32 bits. The MMU provides virtual memory address translation functions, memory control, and error correct code (ECC) functions. The basic processor board clock rate is 170 nanoseconds.

The processor board also contains 64K-bytes of read-only memory (ROM) used for initial program load (IPL) and initial diagnostic functions. Five bipolar gate arrays and miscellaneous TTL logic are used for memory channel and processor channel interface. A bipolar (Random Access Memory (RAM) is used for the reference and change bits associated with virtual memory management support.

The processor board connects to both the memory channel and the processor channel. The memory channel connects only to system memory boards, with the processor channel connecting to both the I/O subsystem and the optional Floating-Point Accelerator.

Memory Channel

The memory channel connects system memory to the processor board. This channel provides a multiplexed address bus, a data bus, and control signals. The address bus supports up to 16M-bytes of system memory. The data bus provides 32 bits of data and 8 bits of error correcting code (ECC). The memory channel is capable of supplying 4 bytes of data every 170 nanoseconds, resulting in a bandwidth of 23.5 megabytes per second.

Processor Channel

The processor channel connects the processor board to the I/O subsystem and the optional Floating-Point Accelerator. This channel provides a multiplexed 32-bit address and data bus, plus control signals to allow access of system memory through the memory management unit by the I/O subsystem. This channel also allows transfer of floating point data and control from the processor board to the Floating-Point Accelerator. External interrupt lines are provided for use by the I/O subsystem.

System Memory Boards

System memory boards plug into the memory channel on the system board and contain the address and buffer logic.

Memory on a board is divided into two banks, with one bank containing only even addresses, and the second bank containing only odd addresses. Interleaving of memory in this manner provides improved memory bandwidth to the processor board. Each memory word contains 32 bits of data and 8 bits of error correcting codes (ECC). The ECC allows correction of single bit errors and detection of all double bit errors. TTL logic is used for address and data buffers on the system memory boards.

Floating-Point Accelerator Board

The optional Floating-Point Accelerator board plugs into the processor channel on the system board and contains the following components:

- NS32081 Floating-Point Accelerator
- Floating point register sets
- Interface and control logic.

A National NS32081 Floating-Point Accelerator is used for floating point applications. Additional interface and control logic on the Floating-Point Accelerator Board allows floating point operations to be executed in parallel with other computations on the processor board.

I/O Subsystem

The I/O subsystem, located on the system board, consists of logic on the system board, and provides the following functions:

- I/O Channel Controller. The I/O channel controller converts the 32-bit processor channel to a standard 16-bit and 8-bit I/O channel.
- System Direct Memory Access (DMA) Controller. The system DMA controller consists of two 8237 modules. They are used to provide 7 independent DMA channels. An eighth DMA channel exists for the optional RT PC 286 Coprocessor.
- Interrupt Controller. Two 8259 modules are used to provide 15 levels of priority interrupts.
- Translation Control Logic. The translation control logic determines the mapping of addresses from the I/O channel to system memory.
- Keyboard/Locator Controller. An 8051 microprocessor and 8255 peripheral interface are used for the keyboard controller. The 8051 microprocessor handles keyboard, locator (mouse or tablet) and speaker communications independent of the processor board.
- Real-Time Clock. A Motorola 146818 CMOS module is used for the real-time clock. The real-time clock is battery-powered and provides both time of day information and nonvolatile memory of certain system configuration information.
- RS232C Serial Port Logic. The IBM 6150 provides logic on the system board for two RS232C serial ports. The IBM 6151 does not provide these serial ports.

I/O Channel

The I/O channel provides a 16-bit and 8-bit data bus for attachment of standard I/O adapters. The IBM 6150 provides six 16-bit slots and two 8-bit slots, with the IBM 6151 providing five 16-bit slots and one 8-bit slot.

System Performance

The RT PC processor board operates at a 170 nanosecond cycle time. Execution time for instructions is contained in Appendix A.

The memory channel can supply 4 bytes of data every 170 nanoseconds, which results in a bandwidth of 23.5M-bytes per second.

System memory refresh operations are controlled by the memory management unit (MMU), and normally occur during idle memory cycles. Typical refresh interference is about one percent of the available memory bandwidth.

System Hardware Overview

The RT PC system hardware includes the system board, processor board, memory boards and Floating-Point Accelerator.

Nomenclature

This section discusses the conventions used on the processor channel and the conventions used on the I/O channel. The I/O Channel Controller (IOCC) provides the connection between the two different conventions.

Notes:

1. MSB = Most Significant Bit
2. LSB = Least Significant Bit

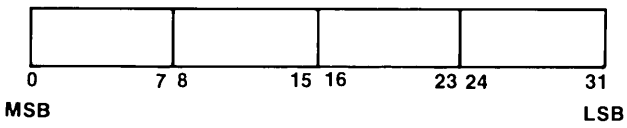
Data Names and Bit Conventions

The RT PC system hardware includes both unique 32-bit system processor features and standard 16-bit I/O channel features. Standard System 370 terminology is used when discussing address quantities, data quantities, and bit numbering related to the 32-bit components. Standard vendor notation is used for the 16-bit I/O channel features.

The following terminology is used for the 32-bit components:

- Word - A word is four consecutive bytes, or 32 consecutive bits.
- Halfword - A halfword is two consecutive bytes, or 16 consecutive bits.
- Byte - A byte is eight consecutive bits.

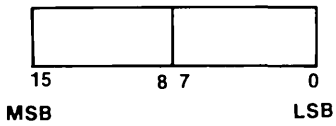
The following bit numbering is used for the 32-bit components:



The following terminology is used when discussing address and data quantities related to the 16-bit I/O channel:

- Doubleword - A doubleword is four consecutive bytes, or 32 consecutive bits.
- Word - A word is two consecutive bytes, or 16 bits.
- Byte - A byte is eight consecutive bits.

The following bit order is used for the 16-bit components:



Hexadecimal Notation

The notation X 'value' is used for all hexadecimal numbers. For example, X'A123' is the 16-bit hexadecimal value A123.

Processor Channel Bit Convention

The processor channel bit convention is as follows:

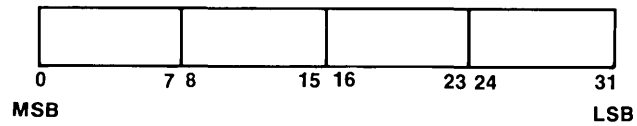


Figure 1-2. Processor Channel Bit Convention

I/O Channel Bit Convention

The I/O channel data bus uses the following convention for 8 bits.

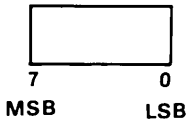


Figure 1-3. I/O Channel 8-Bit Convention

The I/O channel data bus uses the following convention for 16 bits.

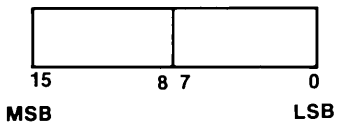


Figure 1-4. I/O Channel 16-Bit Convention

Processor Channel Byte Convention

The processor channel uses the following convention.

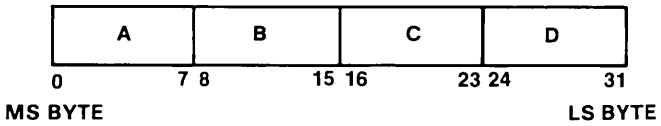


Figure 1-5. Processor Channel Byte Convention

I/O Channel Byte Convention

The I/O channel uses the following convention for a 2 byte (16 bit) channel:

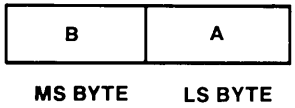
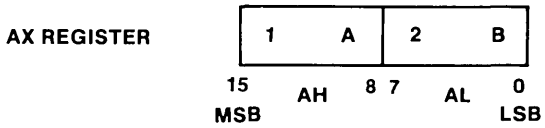


Figure 1-6. I/O Channel Byte Convention

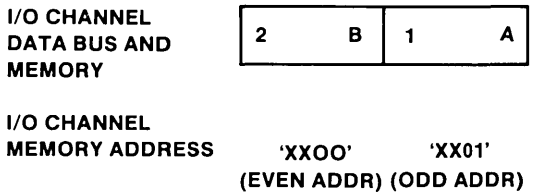
Coprocessor to I/O Channel and Memory Byte Conventions

When the Coprocessor is attached to the I/O channel, its byte reversal characteristics must be considered. Programmers transferring data between the processor channel and the Coprocessor registers must understand this convention.

The following diagram shows the value X'1A2B' in the AX register of the Coprocessor.



When the AX register value X'1A2B'is sent to the I/O channel data bus, it will be as follows:



The Coprocessor stores the least significant byte in the even address and stores the most significant byte in the odd address.

System Memory Conventions

System memory is part of the 32-bit features in RT PC, and uses the following conventions:

Word	A word is four consecutive bytes or 32 consecutive bits.
Halfword	A halfword is two consecutive bytes, or 16 consecutive bits.
Byte	A byte is eight consecutive bits.
Word address boundary	A word address boundary is an address where the two least significant address bits are zero (XX00).
Halfword address boundary	A halfword address boundary is an address where the least significant bit is zero (XXX0).
Even halfword address	An even halfword address is an address where the two least significant address bits are zero (XX00).
Odd halfword address	An odd halfword address is an address where the two least significant address bits are one and zero (XX01).
Even byte address boundary	An even byte address is an address where the least significant address bit is zero (XXX0).
Odd byte address boundary	An odd byte address is an address where the least significant address bit is one (XXX1).

System Memory Addressing Conventions

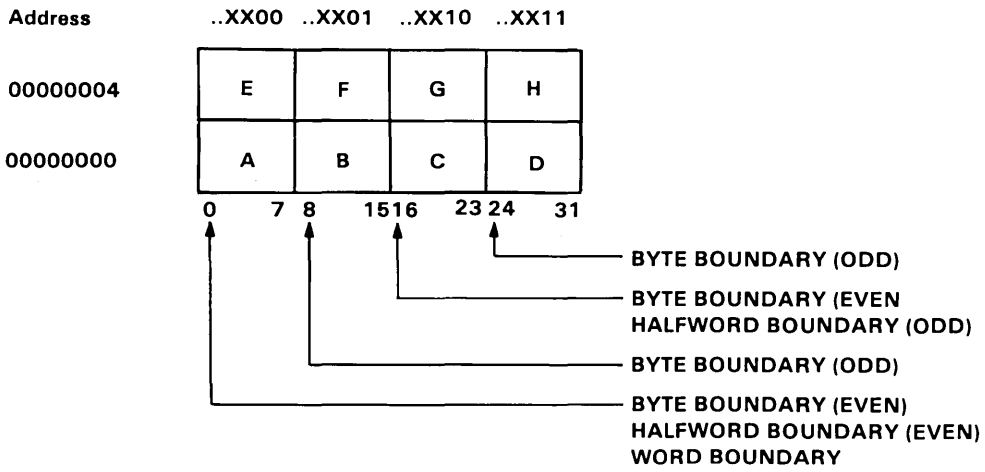


Figure 1-7. System Memory Addressing Conventions

Processor Channel to I/O Address Bus Convention

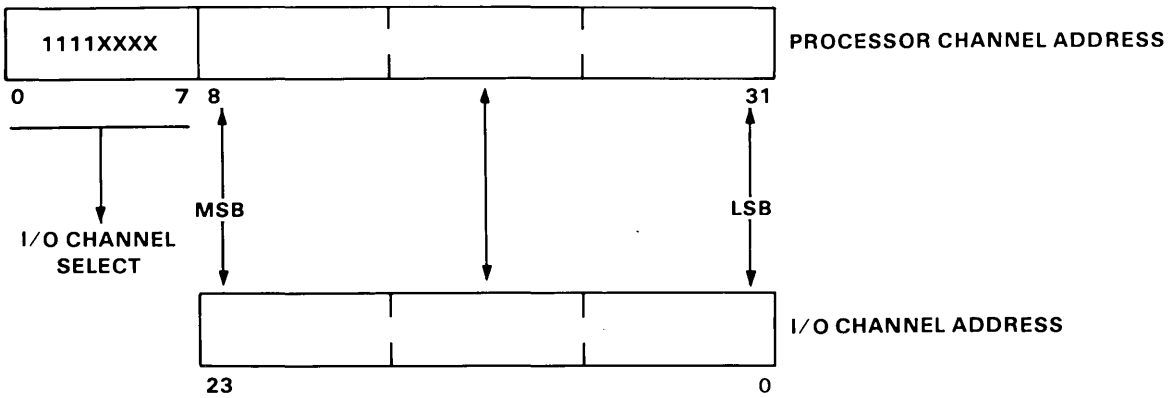


Figure 1-8. I/O Channel Address Bus Mapping

Bits 0-7 of the processor channel select the I/O channel as described in "Addressing Model" on page 1-26. X'F0' selects the I/O channel I/O address space and X'F4' selects the I/O channel memory address space. The low-order 24 bits are used as the I/O channel address.

System Memory to Processor Channel Byte Positions

The following figure shows data movement between system memory and the processor channel.

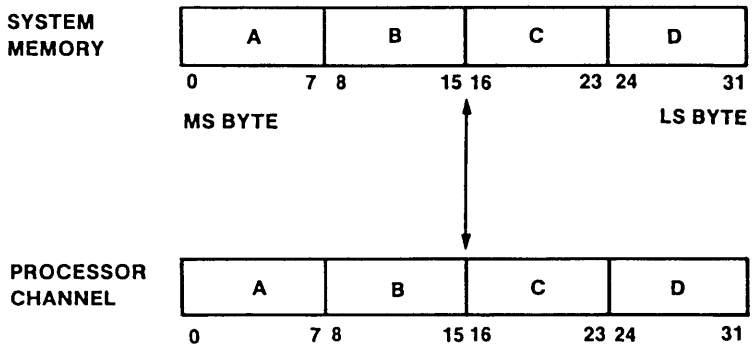


Figure 1-9. System Memory to Processor Channel Bytes

The IOCC always reads a word from system memory. Byte A is always aligned on a word address boundary. The processor channel contents are the same for the following:

- A word read
- An even halfword read (AB)
- An odd halfword read (CD)
- Any single byte read.

Processor Channel to I/O Data Bus Conversion

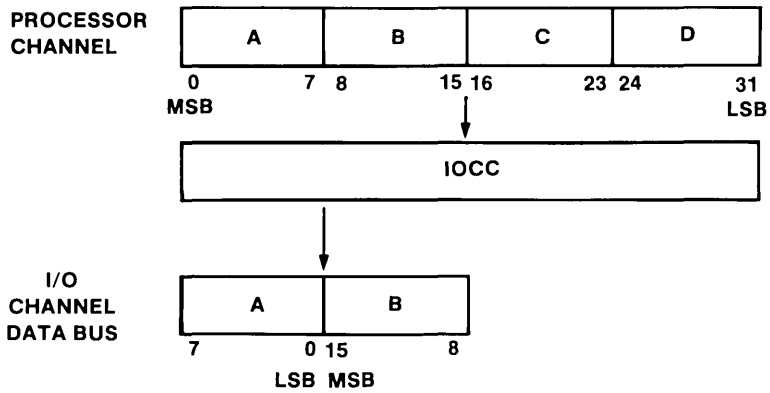


Figure 1-10. Processor Channel to I/O Data Bus Mapping

For a 16-bit transfer, the IOCC receives byte A (bits 0-7) from the processor channel and sends them to byte A (bits 7-0) on the I/O channel data bus. Simultaneously, the IOCC receives byte B (bits 8-15) from the processor channel and sends them to byte B (bits 15-8) on the I/O channel data bus. The actual I/O channel data bus format is as follows.

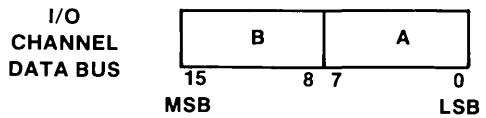


Figure 1-11. I/O Channel Data Bus

If processor channel bytes C and D were also requested, the IOCC would multiplex them onto the 16-bit I/O data bus. The IOCC sends processor channel byte C (bits 16-23) to I/O channel data bus A (bits 7-0) and processor channel byte D (bits 24-31) to I/O channel data bus byte B (bits 15-8).

IOCC to I/O Channel Alignment

The I/O and memory maps on the I/O channel are a collection of 8-bit (byte) addressable elements. There are 16,777,216 such addressable elements within each map. Each transaction over the RT PC I/O Channel (Programmed Input/Output or Direct Memory Access) is capable of transferring 2 bytes (16 bits) of data. The number of bytes actually transferred over the channel during one channel cycle is determined jointly by both the adapter initiating the operation and the target adapter. The transfer defaults to a single byte if one of the adapters is an 8-bit adapter. Transfers of more than 16 bits are accommodated by multiple I/O channel operations.

Both the processor channel and I/O channel have control signals which indicate the amount of data transferred during the current transaction. Depending on the source initiating the transaction, one set of control signals dictates the transfer size while the other set will simply reflect that requirement.

On all system processor PIO operations, the actual data pattern transferred over the I/O channel and the number of channel cycles required is dependent on the amount of data transferred and the width of the selected I/O adapter. This means if the number of bytes transferred by one system processor instruction is larger (2 or 4 bytes) than the data width (1 or 2 bytes) of the selected I/O adapter the IOCC automatically performs multiple I/O operations. For example, a 32 bit microprocessor PIO operation to:

- An 8-bit wide adapter requires the IOCC to perform four distinct channel cycles with each cycle transferring 8 bits of data.
- An 16-bit wide adapter requires the IOCC to perform two distinct channel cycles with each cycle transferring 16 bits of data.

Boundary alignment is strictly enforced on all transfers through the IOCC. The low order address bits are modified by the IOCC so the resulting address conforms to the definitions stated in "System Memory Conventions" on page 1-18. The IOCC forces address bits A(1,0) to 00 on all 32-bit transfers through IOCC for system processor PIO. Similarly, the IOCC forces address bit A(0) to 0 on all halfword transfers through IOCC.

On the I/O channel a 2-byte transfer is flagged by the 'Byte High Enable' (SBHE) signal. The selected I/O adapter signals its data bus width through the 'IOCS16' or 'MEMC16' signal for a 16 bit adapter to either the I/O map or memory map respectively. On the processor channel, the transfer size information is encoded within 'DAB 2, 3' of the address cycle with data being positioned according to the value of the address bus. For a I/O channel byte transfer the two least significant address bits determine which one of the four bytes within the processor channel word participates in the transfer. Also for I/O channel word operations, the second least significant address bit determines which of the two halfwords within the processor channel is selected (with the least significant address bit not used).

The mapping within the IOCC of 8- and 16-bit adapters onto the 32-bit processor channel is accomplished through the multiplexer network on the I/O channel. The IOCC positions data correctly on the I/O channel for system processor PIO operations for 8- and 16-bit adapters on the

initial I/O channel cycles based solely on the low order address bits and the transfer size. If the adapter's response indicated a width less than the transfer size then subsequent cycles are required. For these multiple cycles, the multiplexer control is a function of the cycle and adapter width. The IOCC activates the 'SBHE' signal for 16 and 32 bit PIO transfers. This signal is not changed through the duration of the complete operation (1, 2, or 4 I/O channel cycles).

Data alignment through the IOCC, and on both the processor channel and I/O channel, is independent of the device location initiating the operation (I/O or processor channel attached). Alignment is dependent only on the address and size of the transfer. Data is aligned identically for 16-, and 8-bit data flowing from the processor channel to the I/O channel on either a system processor write or DMA read. Data is also aligned identically for 32-, 16-, and 8-bit transfers flowing from the I/O channel to the system processor on either a I/O DMA write or a system processor read.

Programming Note

The IOCC permits the programmer to perform a doubleword (32 bit) read or write to a byte wide adapter on the I/O channel. The IOCC converts this single transaction into either 2 or 4 I/O channel operations depending on the amount of data transferred and the characteristics of the selected adapter (16 or 8 bits). If the operation is directed to the memory map on the I/O channel, the IOCC increases the address between consecutive channel transfers. If the operation is directed to the I/O map on the I/O channel, the IOCC holds the address constant between consecutive channel transfers.

This facility can not be used without knowledge by the programmer of the characteristics of the addressed adapter. In particular, the adapter must be capable of handling back-to-back operations without violating any timing restrictions it may have. The responsibility for knowing the characteristics of the I/O adapter and not violating these restrictions resides with the programmer.

If a I/O adapter requests one byte (byte D), the processor channel presents the byte as shown in Figure 1-7 on page 1-19. The IOCC aligns the byte as shown below. This allows the adapter to get the byte in one cycle according to its data width.

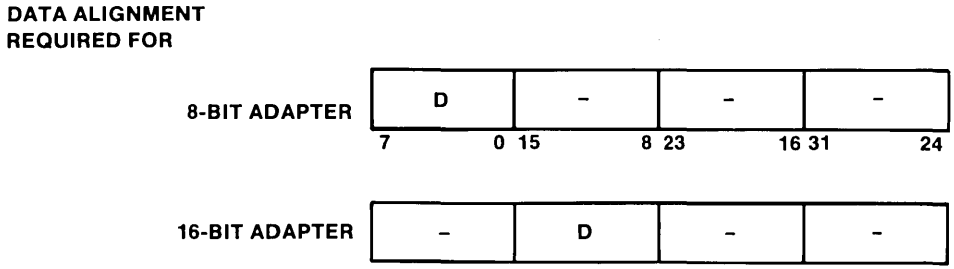


Figure 1-12. Data Alignment for 8- and 16-Bit Adapters

Addressing Model

This section describes the addressing model of the RT PC system board. The addressing model is described from the view that the IOCC is attached to the processor channel.

This section further describes how the addresses move through the IOCC and what action the IOCC will take on addresses generated from both the processor channel and I/O channel. Detailed descriptions of hardware implementation found in other sections of this document are referred to throughout this section.

The nomenclature used in this section for bit numbering and data naming are described in "Nomenclature" on page 1-13.

The system processor communicates with the system through the processor channel. The processor channel has both I/O and memory address space (maps) that are accessible from either the I/O channel or the system processor. The memory map on the processor channel is further divided into real and virtual accesses. The I/O Channel Controller (IOCC) is a device on the processor channel. The system processor has access to the I/O adapters through the IOCC and, conversely, I/O adapters have access to the system memory. The I/O channel has both I/O and memory maps. The following two sections describe the system processor's view of the I/O channel maps and an adapter's view of both the I/O channel and processor channel maps.

The term segment is used in the following sections to describe various address spaces on the processor channel. The most significant four bits of the 32-bit address on the processor channel select one of sixteen areas, which are called segments. These segments are numbered zero through fifteen and are called segments 0 through F. Segment F is assigned to the IOCC and is further decoded in the IOCC to select between the I/O channel I/O map and the I/O channel memory map as described in "I/O MAP" on page 1-28 and "Memory Map" on page 1-29.

Addressing From the System Processor

This section covers access to both of the address maps on the I/O channel as viewed from the program executing system processor instructions. The view that the system processor has of these address maps is shown in Figure 1-13 on page 1-30.

The RT PC system configuration has four devices attached to the processor channel. These devices, along with their characteristics, are:

System Processor

The system processor can initiate an operation over the processor channel through the execution of the following commands:

- I/O read and write (IOR and IOW) instructions to control registers, translation buffers, and segment registers in the memory management unit.
- Load or store instructions to system memory, IOCC control registers, the I/O channel, or the floating point accelerator.

Use of the IOR and IOW instructions is described in "Input/Output" on page 11-83 and "Control Registers" on page 11-113. The various load and store instructions are described in "Load and Store Instructions" on page 11-31.

The system processor does not have any registers that are addressable over the processor channel. Thus, it never responds to processor channel requests.

Memory Management Unit

The memory management unit (MMU) cannot initiate a processor channel operation. That is, it cannot generate an address. The memory management unit will either supply data in response to a read request or will receive data on a write operation. It responds to I/O space (IOR, IOW) operation for control functions and memory space (Memory Read, Write) for memory references.

Floating-Point Accelerator Option

The Floating-Point Accelerator cannot initiate a processor channel operation. That is, it cannot generate an address. The Floating-Point Accelerator is always the target of an address. Thus, it will either supply data in response to a read request, or it will receive data on a write operation. It responds to memory space (Memory Read,Write) operations for referencing segment 'FF' of the address map.

IOCC

The IOCC can both initiate operations on the processor channel as well as be the target of processor channel operations. IOCC initiates processor channel operations as a result of a DMA operation on the I/O channel. Furthermore, it is the target of a processor channel operation resulting from LOAD and STORE instructions executing in the system processor or from a system processor instruction fetch. The IOCC responds only to processor channel address cycles which are referencing segment 'F0' or 'F4' of either the real or virtual address map. All references to the I/O map on the processor channel are ignored.

For system processor access to the I/O channel (to either the I/O map or the memory map) the IOCC performs the number of I/O cycles (1,2, or 4) required to transfer the data. This is possible because the I/O adapter identifies its data width when selected. The IOCC first aligns the address to the appropriate boundary as indicated by the size of the transfer (for instance, forcing word or double word) address by modifying address bits A(1) and/or A(0) as required. Since boundary

alignment is strictly enforced, the address used for the second and subsequent channel cycles will differ from the first by the 2 low order address bits. Refer to "System Memory Conventions" on page 1-18.

The IOCC supports system processor access to the I/O channel address map for both data (Memory Read, Write) and instructions.

The rest of this section discusses access to the I/O map and memory map of the I/O channel. Refer to Figure 1-13 on page 1-30 and Figure 1-15 on page 1-34.

I/O MAP

Access to the I/O address map of the I/O channel is through the first 16 M-bytes of the segment 'F'. Access to this map is controlled by the state of the I/O Map Access Authority (IMAA) latch.

Access attempts to the I/O address map of the I/O channel in unprivileged state with the IMAA set to privileged state are not allowed by the IOCC. A protection violation error will be reported. Refer to the "Channel Status Register (CSR)" on page 5-63. Access to the I/O address map of the I/O channel in privileged state is allowed regardless of the state of the IMAA.

Note: Access to the channel status register (CSR) within the IOCC is restricted to privileged state only.

The IOCC supports byte (8 bits), word (16 bits), and doubleword (32 bits) access to the I/O address map. The IOCC activates a control signal on the I/O channel to indicate the transfer size as stated by the system processor instruction executed. If the I/O adapter is not capable of transferring the requested number of bytes, multiple accesses are made to the same address until the requested amount of data has been transferred.

One characteristic of the system processor structure is that non-adjacent (order of execution) Read/Write instructions to external devices (memory management unit or IOCC) can (and often do) arrive at the device as if they were adjacent. That is, the programmer cannot guarantee the time period between consecutive accesses to an adapter simply by inserting some instructions between the two I/O channel references. The effect of this is that the I/O adapter may receive I/O commands at a rate faster than it can process them. The IOCC is not capable of ascertaining any performance limitations relative to 'wait' time between consecutive I/O operation associated with a given address in the I/O space. Thus, the responsibility for knowing the characteristics of the I/O adapter and not violating these restrictions resides with the programmer.

To aid software in generating wait time for those adapters that require a minimum time delay between consecutive accesses, a 'NOP like' I/O command has been provided that can be inserted between the I/O instructions to the adapter. Access to this I/O delay register between otherwise sequential accesses to I/O subsystem and I/O channel adapter options prevents the possible loss of data. The system board devices which should use this mechanism are: 8237s, 8259s, real time clock (RTC), and the keyboard adapter.

Boundary alignment is strictly enforced on the I/O channel. That is, the IOCC may alter the low order address bits as received from the processor channel before they are applied to the I/O channel. Refer to "IOCC to I/O Channel Alignment" on page 1-23.

Memory Map

Access to the memory address map of the I/O channel is through to the fifth 16M-bytes of segment 'F'. Access to this map is controlled by the state of the Memory Map Access Authority(MMAA) latch.

Access attempts to the I/O channel memory address map in unprivileged state with the MMAA set to privileged state are not allowed by the IOCC, and a "protection violation" error will be reported. Access to the memory address map of the I/O Channel in privileged state is allowed regardless of the state of the MMAA.

The IOCC supports byte (8 bits), word (16 bits), and doubleword (32 bits) access to the memory address map. The IOCC activates a control signal on the I/O channel to indicate the transfer size as stated by the system processor instruction executed. If the I/O adapter is not capable of transferring the requested number of bytes, multiple access will be made to consecutive addresses until the requested data has been transferred. However, since none of the I/O subsystems use any portion of the memory address map, this function is not supported on the system board.

Boundary alignment is strictly enforced on the I/O channel. That is, the IOCC may alter the low order address bits as received from the processor channel before they are applied to the I/O channel. Refer to "IOCC to I/O Channel Alignment" on page 1-23.

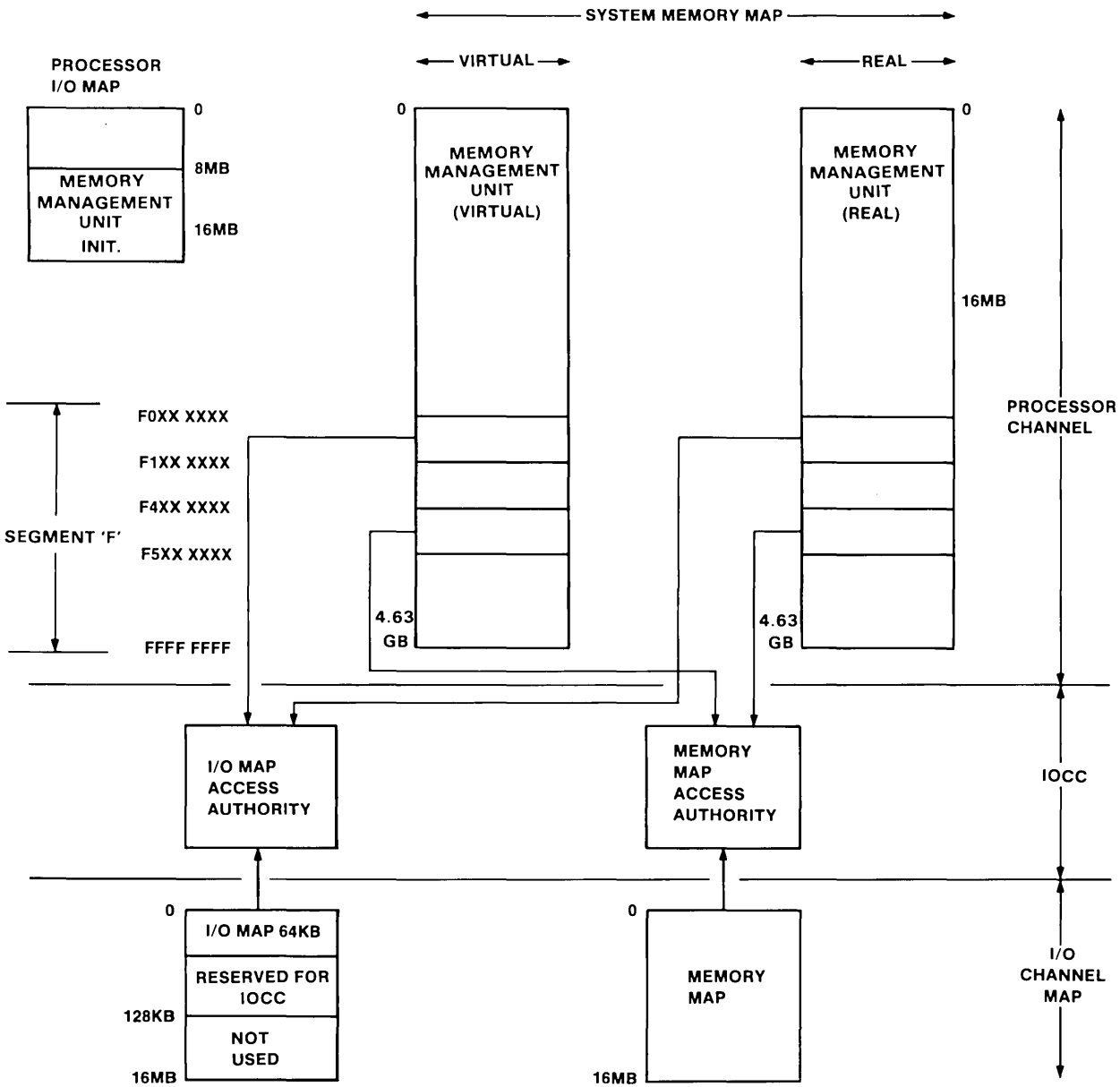


Figure 1-13. System Processor Access to Address Maps. This diagram shows the view of the I/O channel, I/O and memory maps from the system processor and the access authority control.

Addressing From the I/O Channel

This section covers access to the maps on the I/O channel and access to system memory as viewed from an adapter on the I/O channel. The view that a alternate controller has of memory and its access path is shown in Figure 1-14 on page 1-33.

The IOCC responds to addresses originating on the I/O channel. I/O addresses are restricted to the I/O subsystem facilities. The I/O address is not propagated onto the processor channel. Access to the I/O subsystem by I/O adapters is controlled by three channel access control bits. Memory addresses to which the IOCC will respond are under program control. The IOCC does not contain any facilities within the memory map, but it will respond to the I/O channel operation for the processor channel.

I/O MAP

An I/O adapter cannot access the I/O address map on the processor channel. However, it can access the other adapters on the I/O channel in the I/O map.

Most of the registers internal to the IOCC can be accessed from an alternate controller adapter operating in region mode. These registers are in the I/O map and are addressed with the IOR or IOW control lines. Access to this I/O map from the system processor can be controlled through the IMAA latch. A separate set of control latches must be used to control access by alternate controller adapters.

Note: Alternate controller adapters cannot address (and, therefore, cannot access) the channel status register (CSR) or the translation control word (TCW) RAM.

Memory Map

The I/O channel architecture of RT PC permits a variety of adapter types to exist in the memory map:

- Memory board options
- I/O adapters with RAM addressable in the memory map
- I/O adapters with internal registers addressable in the memory map
- IOCC which permits access to system memory by alternate controller adapters.

RT PC system architecture requires the physical location of Random Access Memory (RAM) in the I/O memory map be transparent to the alternate controller adapter referencing it. For example, if performance of the 286 Coprocessor is important, then RAM should be located on the I/O channel. However, if low entry cost is desired, the 286 Coprocessor can execute out of system memory. The

application program is to be unaware (except for time dependent code) of the physical location of the memory out of which it is executing.

When a memory address is placed on the I/O channel, it is examined by all adapters, including the IOCC, to determine if the address is within the range covered by the adapter. If a match is found by one adapter, it responds with the appropriate channel control signals. To maintain the integrity of the data and to avoid damage to the channel interface circuitry, it is important that only one adapter responds to a given address.

One of the primary functions of the IOCC is to provide access to system memory. Thus, the IOCC must examine each reference to the memory map on the I/O channel to determine if the address was intended for system memory. The effect is that this requirement places an unusual burden on the IOCC not commonly found in I/O adapters. In general, I/O adapters have a fixed address range in the memory map to which they respond. The address range to which the IOCC responds is variable since it can differ for alternate controller and DMA devices as well as vary with application. This selection is controlled by the values written into the IOCC translation mechanism by the software. Refer to the section on IOCC translation mechanism.

All addresses in the memory map applied to the I/O channel by adapters are processed by the IOCC under control of its TCW. This translation mechanism is a programmable facility controlling target memory location, memory protection, defining access mode, and reformatting the address. The central part of this mechanism is a 1K x 16 bit RAM.

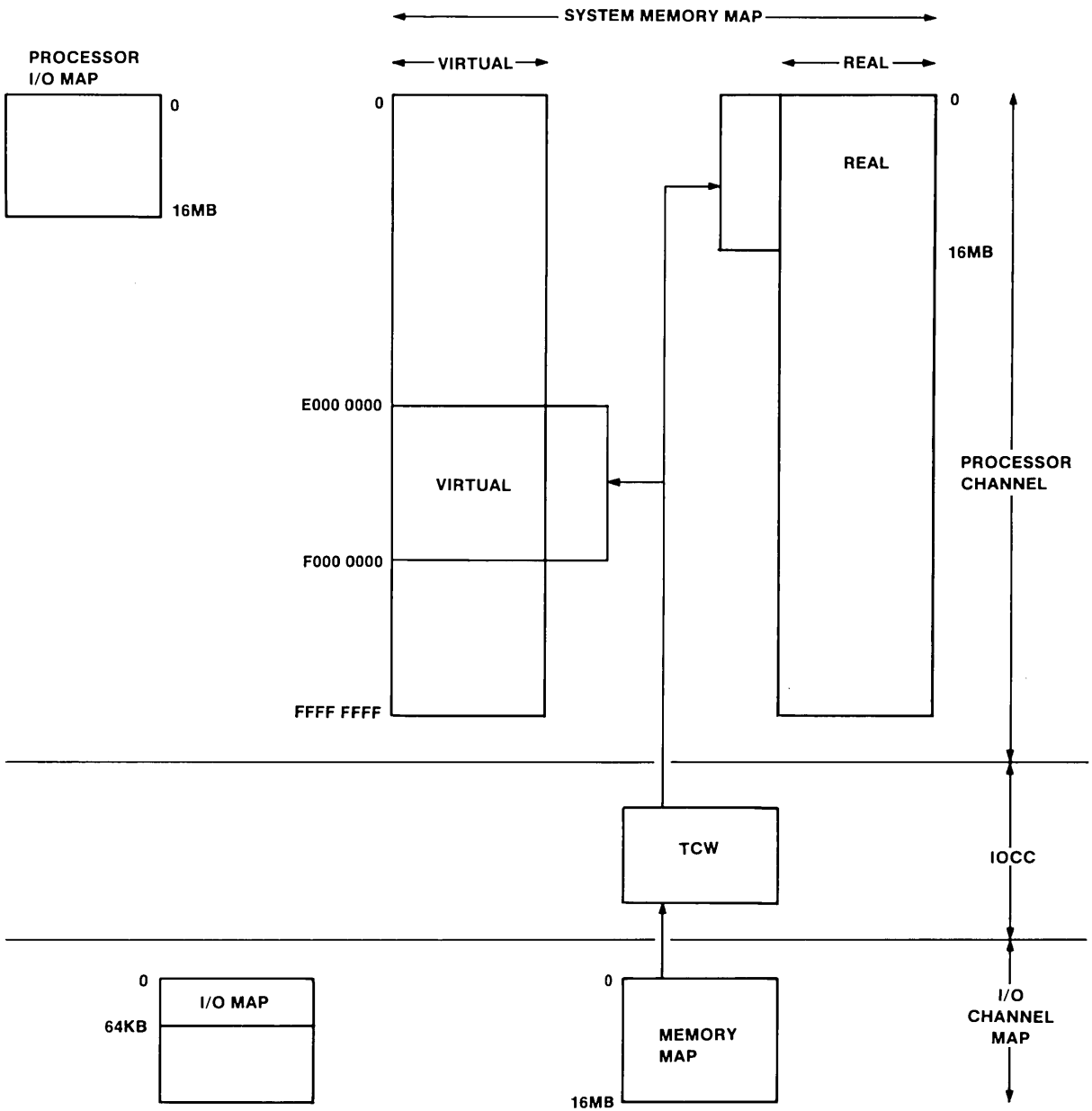


Figure 1-14. Access to System Memory Maps from IOCC. This diagram shows how an I/O channel adapter views the system memory map IOCC address translation.

System Processor Address Map

The system processor can directly address two completely independent address spaces. The address spaces are called memory and I/O. Access to these maps by the system processor is controlled by the instruction being executed (I/O or Memory Load/Store) and memory mode (real or virtual). The I/O address is 24 bits, while the system address is 32 bits. The address maps from the perspective of the system processor are given in Figure 1-13 on page 1-30 and Figure 1-14 on page 1-33.

Note: Even though the system processor supports a 32-bit real memory address, the RT PC system is limited to a 24-bit real memory address.

Address Range	Size	Device
00 0000 - 7F FFFF	8 MB	Not Defined
80 0000 - 80 FFFF	64 KB	Processor Channel Device Initialization
81 0000 - 81 FFFF	64 KB	Memory Management Unit
82 0000 - FF FFFF	8064 KB	Not Defined

Figure 1-15. System Processor Address I/O Map

Address Range	Size	Device
0000 0000 - 00FF FFFF	16 MB	Memory Management Unit
0100 0000 - EFFF FFFF	--	Not Defined
F000 0000 - F0FF FFFF	16 MB	I/O Channel I/O Map
F100 0000 - F3FF FFFF	48 MB	Reserved
F400 0000 - F4FF FFFF	16 MB	I/O Channel Memory Map
F500 0000 - FEFF FFFF	158 MB	Reserved
FF00 0000 - FFFF FFFF	16 MB	Floating Point Accelerator

Figure 1-16. System Processor Real Memory Address Map

Address Range	Size	Device
0000 0000 - EFFF FFFF	4 GB	Memory Management Unit
F000 0000 - F0FF FFFF	16 MB	I/O Channel I/O Map
F100 0000 - F3FF FFFF	48 MB	Reserved
F400 0000 - F4FF FFFF	16 MB	I/O Channel Memory Map
F500 0000 - FEFF FFFF	158 MB	Reserved
FF00 0000 - FFFF FFFF	16 MB	Floating-Point Accelerator

Figure 1-17. System Processor Virtual Memory Address Map

Interrupts

The interrupt facility permits the system processor to change its status at the request of some other system element. The interrupting sources are:

- Software generated interrupt request from system processor's interrupt request buffer
- System component on the processor channel
- I/O adapters on the I/O channel.

This request is generated when an I/O adapter reaches some condition established by software or the occurrence of some error condition. All interrupts are fielded and processed by the system processor. The interrupt processing consists of saving the current program status and establishing the program status for the routine servicing the interrupt. Interrupt handling by the system processor is described in "Interrupts" on page 11-19.

Interrupt Levels

The system processor supports 5 levels of external interrupts. Normally, interrupts are serviced in a fixed priority sequence with interrupt level 0 having the highest priority and level 4 having the lowest. The I/O channel supports 11 levels of interrupt request. The I/O subsystem uses 4 levels in addition to those on the I/O channel.

The IOCC interrupt controller (refer to "Interrupt Controllers" on page 5-56) maps the 11 I/O channel interrupts and 4 system board interrupts onto two system processor interrupt levels. It also provides the ability to mask off individual interrupt signals.

System Processor Trap

The system processor interrupt mechanism supports a second class of interrupts called the I/O Trap. Activation of the 'I/O Trap' signal causes the system processor to do a program status word (PSW) exchange to the machine check level. The memory management unit activates the 'I/O Trap' signal upon detection of uncorrectable ECC errors (or, optionally, on all ECC errors) or upon detection of an 'Early Power-Off Warning' signal from the power supply.

Interrupt Level Assignment

The assignment of the five system processor interrupt request signals and the trap signal are as follows:

Interrupt Level	Assignment
0	System Attention Interrupt Sequence from Keyboard
1	Real Time Clock Interrupt
2	IOCC Errors
2	Memory Management Unit Program Check
3	I/O Channel Levels
4	I/O Channel Levels
Trap	Memory Management Unit Machine Check
Trap	Early Power Off Warning

Figure 1-18. Interrupt Level Assignment

System Processor Interrupt Processing

The system processor accomplishes the interrupt handler function through a combination of:

- Hardware within the IOCC
- Interrupt control logic within the system processor
- Software executing in the system processor.

The interrupts are fielded directly by the system processor. If the interrupt mechanism within the system processor is enabled, and the interrupt request is at a higher level than is currently being processed by the system processor, a PSW exchange is initiated. If the new interrupt level can be reached by multiple external events, the interrupt service must then determine the interrupting source. If the new level is one assigned to the I/O channel, the interrupt service routine must first read the interrupt controller to determine the active I/O channel interrupts and, second, must read the adapter's status register.

Note: The information format of the status registers, and the addresses of these registers, are adapter dependent.

The interrupt routine issues the required commands to the adapter to turn off the interrupt signal, and it executes on the 'interrupt ' level. The routine's last task is to return control to the interrupted level. Further processing of the interrupt request is done on a lower interrupt level, permitting other I/O interrupts to be serviced.

Once the source and reason for the interrupt has been identified, the interrupt service program issues the required instructions to satisfy the condition within the I/O adapter. This routine normally executes on an interrupt level of lower priority than that on which the adapter was interrupted.

The interrupt service program must be aware that multiple I/O adapters may have active interrupt requests pending at the same time. Thus, satisfying the requirements of one adapter may not be sufficient to cause the interrupt request (I/O channel signal) to go inactive. This condition is likely to occur on I/O channel levels. To save time, the interrupt service routine reads the interrupt controller before exiting the interrupt level.

Section 2. Processor Boards

CONTENTS

About this Section

This section contains information about the processor boards used by the IBM RT PC system and how they interfaces with the rest of the system. Included in this section is a description of the processor channel signal lines and the memory channel signal lines.

Processor Board Description

The RT PC processor board includes an IBM designed, custom logic proprietary 32-bit microprocessor and memory management unit (MMU). Features of these VLSI chips follow.

System Processor

The system processor is a single chip microprocessor that is mounted in a 175-pin package. The processor includes:

- A 32-bit processor on a single chip
  - Very Large Scale Integration (VLSI) custom logic design
  - A 32-bit address and data quantities
  - Single cycle execution of most register-to-register instructions
  - High memory bandwidth for CPU and I/O
- High performance virtual memory support
- Privileged and unprivileged states for system integrity
- A 32-bit timer facility
- Extensive error checking including:
  - Power On Reset (POR) microcode diagnostic
  - Programming errors (illegal instructions, invalid addresses)
  - Hardware errors (parity errors, timeout).

Detailed information on the system processor is provided in Section 11.

Memory Management Unit (MMU)

The memory management unit implements a single level memory address translation architecture. The MMU contains all hardware required for address translation, translation lookaside buffer (TLB) management, and automatic reload on TLB miss.

The MMU is packaged in a 175-pin package. The MMU features include:

- Interfaces to the 32-bit microprocessor
- Performs virtual address translation

  - A 40-bit virtual address
  - Supports 2K and 4K page sizes
  - Provides internal ECC logic for system memory
- Provides memory control to external system memory and IPL ROM
- Supports up to 16M-bytes of system memory
- Provides control of and access to reference and change bits.

Detailed information on the memory management unit is provided in Section 11.

Reference/Change (R/C) Array

The R/C array is a 16K x 1 static RAM. The R/C array stores the reference and change bits associated with pages in memory. Access and use of the R/C array is described in "Reference And Change Bits"on page 11-111.

System Initial Program Load (IPL) ROM

The system IPL ROM contains basic hardware tests which are run everytime the hardware is turned on. Functions provided by the system IPL are described in "System IPL ROM" on page 7-4.

Processor Board Function Overview

The processor board, as depicted in Figure 2-1 is divided into two major functional units and two interfaces. The system processor and the memory management unit are the two major components on the board. Six additional IBM custom designed modules are used on the processor board to create the external interface.

The two processor board interface data flow blocks are shown in Figure 2-1.

"Processor Channel Interface" on page 2-8 describes the Processor Channel Interface.

"Memory Channel Interface" on page 2-16 describes the Memory Channel Interface.

The processor board interfaces are shown in Figure 2-2 on page 2-7.

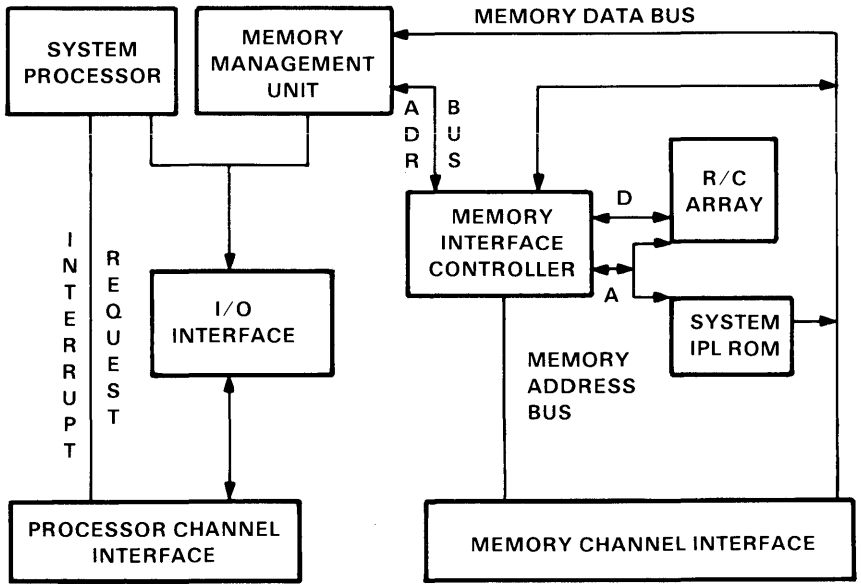


Figure 2-1. Processor Board Data Flow

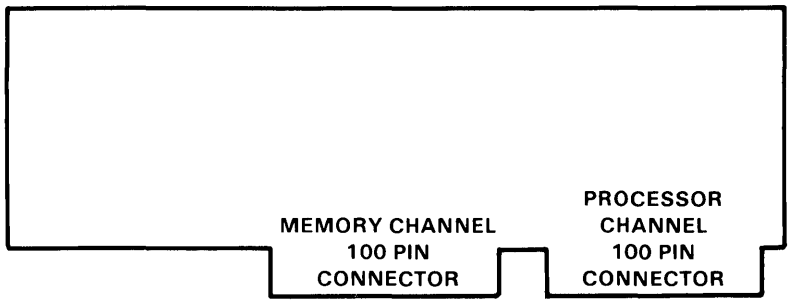


Figure 2-2. Processor Board Interfaces

Processor Channel Interface

The processor channel interface connects the RT PC processor board with the I/O subsystem and the Floating-Point Accelerator slot. It is a multiplexed 32-bit interface with appropriate signals to control programmed I/O (PIO) and direct memory access (DMA) sequences. Miscellaneous processor control lines such as interrupts and power on reset are also part of this channel section.

Common lines on the channel connect to both the IOCC and the Floating-Point Accelerator (FPA) board. Processor board input control lines include resistive pullups and may be driven with open collector drivers.

The processor channel interface is illustrated in Figure 2-3 on page 2-9.

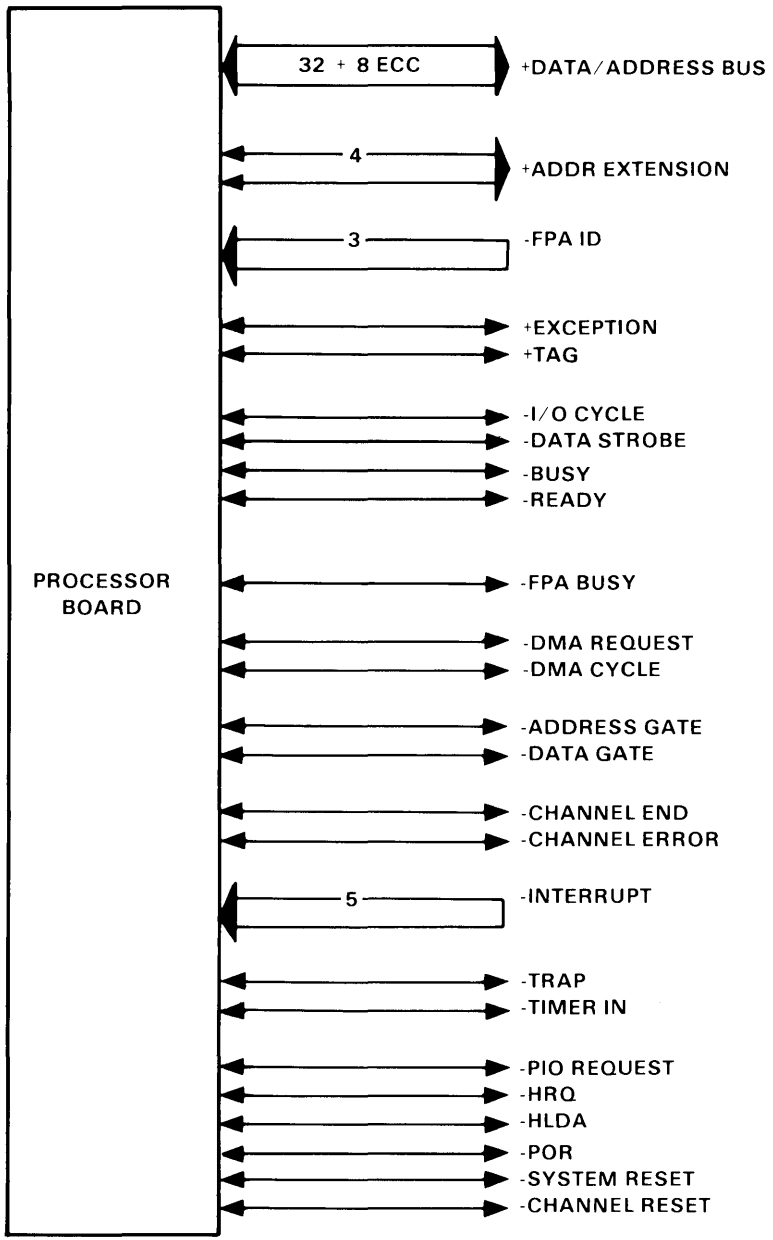


Figure 2-3. Processor Channel Interface

Figure 2-4 illustrates connection of the processor board, IOCC, and the Floating-Point Accelerator slot.

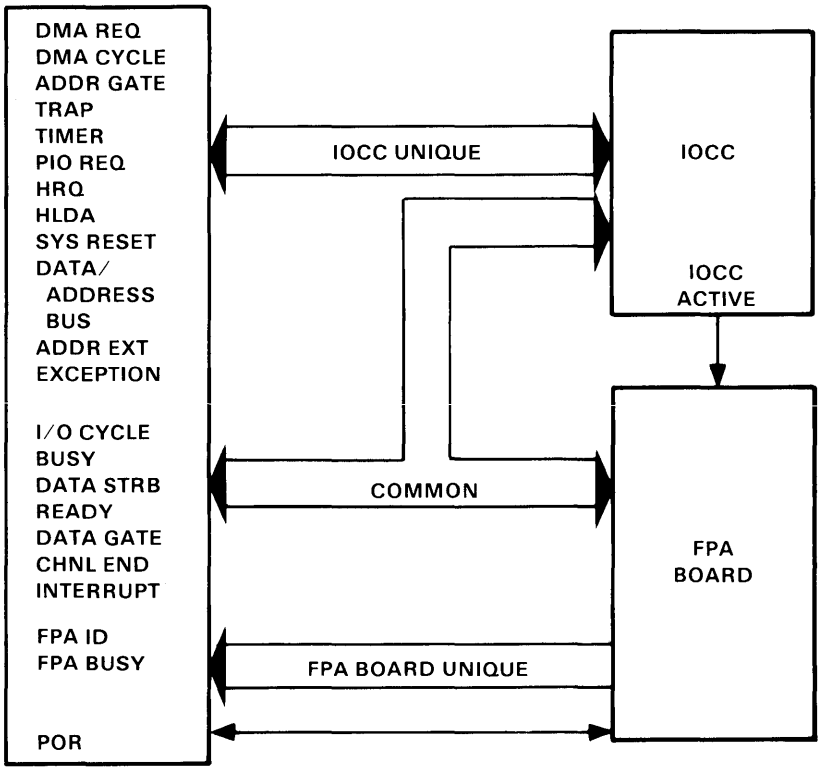


Figure 2-4. Processor Channel Interconnections

Signal Line Definitions

This processor channel interface consists of 73 signals and includes the following signal lines.

Data Address Bus

The 'data address bus' (DAB) is used to transmit address, data, and control between the processor and the I/O subsystem or Floating-Point Accelerator board.

Refer to Figure 2-5 for the data address bus format.

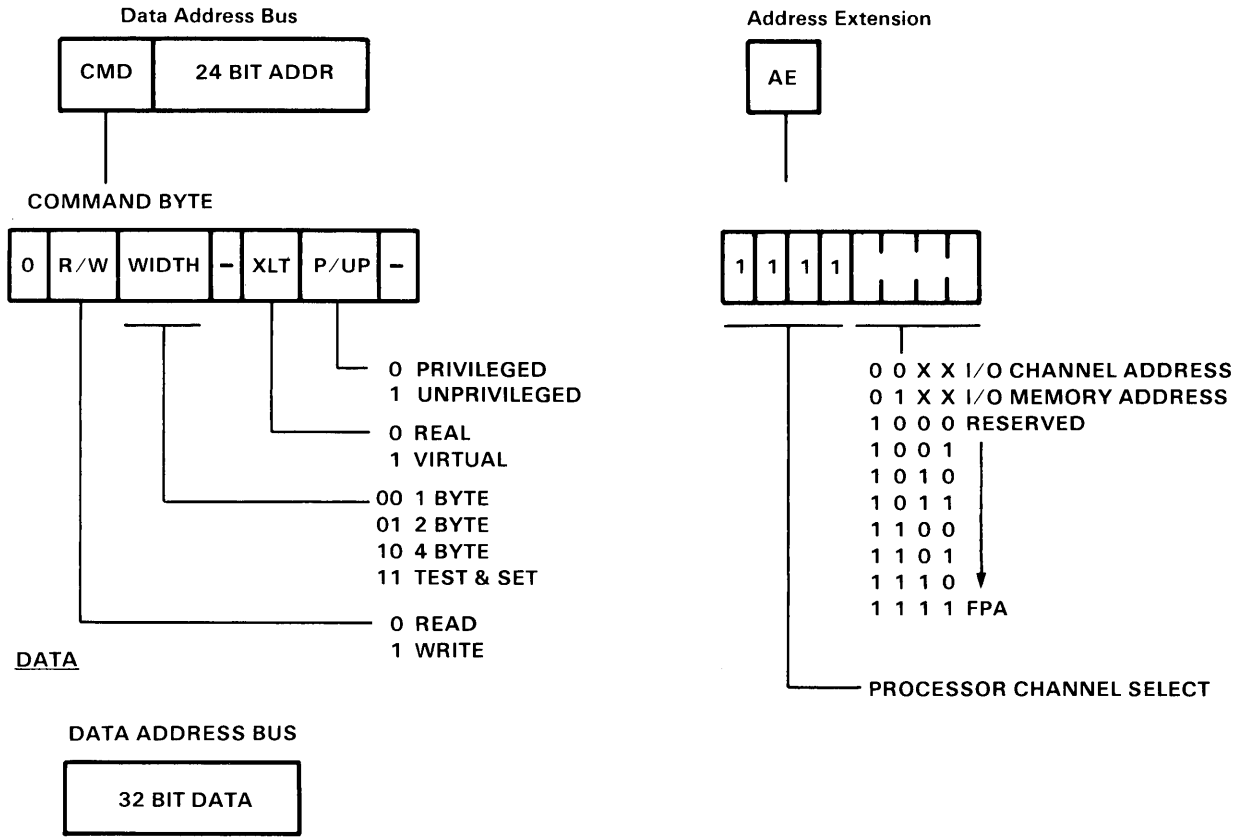


Figure 2-5. Data Address Bus Format

Address Extension

The 'address extension' provides subaddressing or control functions within the IOCC or Floating-Point Accelerator board devices.

+ Exception

'Exception' is an error response to PIO commands or DMA requests and indicates the operation cannot be executed or did not execute correctly.

+ Tag

The processor channel allows up to two PIO commands to be executing concurrently; two commands are not required to complete in the same order they were initiated. To link commands and (completion) replies, a 'tag' bit is supplied when a command is issued. This 'tag' is to be returned along with the return status to indicate the PIO command currently completing.

-I/O Cycle

The 'I/O cycle' signal is activated whenever a PIO command is accepted.

-Data Strobe

'Data strobe' is used as a clock to signal that valid address or data is on the data bus.

-PIO Request

The 'PIO request' signal is activated on queued PIO commands to I/O channel devices or I/O channel memory to signal the arbiter that the Coprocessor should be dearbitrated.

-Ready

The 'ready' signal indicates completion of channel operations.

-Busy

The 'busy' signal holds I/O cycle active. While active, no further (nondiagnostic) PIO commands are accepted from the processor.

-FPA Busy

The 'FPA busy' signal causes a (busy) rejection of PIO commands directed to the Floating-Point Accelerator slot.

-FPA ID

The 'FPA id' signal indicates which address combination the FPA responds to.

-DMA Request

The 'DMA request' signal initiates a DMA memory cycle.

-DMA Cycle

The 'DMA cycle' is the response to a DMA REQUEST.

-Addr Gate

'Addr gate' controls tri-state drivers within the IOCC gating the I/O channel address onto the data bus.

-Data Gate

The 'data gate' signal controls tri-state drivers within the IOCC and Floating-Point Accelerator board for gating data or exception status onto the data bus.

-Channel Error

The 'channel error' signal is activated when a transmission error occurred from the IOCC or Floating-Point Accelerator board to the system.

-Interrupt

These 5 '-interrupt' signals are wired to the processor interrupt inputs for levels 0 - 4.

-Trap

The '-trap' signal is wired to the processor 'trap' input and may be used for functions such as early power-off warning. The 'trap' signal is edge sensitive.

-System Reset

The '-system reset' signal resets the system processor, and produces a 'channel reset' when the processor finishes its diagnostics. System memory is not disrupted by this signal.

-Channel Reset

The '-channel reset' signal is activated whenever a 'channel reset' is issued. This normally occurs on power up or after a processor error timeout.

-POR

'Power-on reset' is active during power on and is deactivated (100-500 ms) following all voltages becoming valid.

-HRQ

The '-HRQ' signal is activated by the IOCC arbiter to request usage of the IOCC data path and the I/O channel.

-HLDA

'HLDA' is the processor response to HRQ.

-Channel End

The '-channel end' signal is activated following presentation of ready. The low to high transition of this signal may be used as an indication of the end of the command.

-Timer In

The 'timer in' signal drives the processor timer.

Arbitration

The various operations (PIO, DMA) performed or controlled by the IOCC are competitive in nature and require shared usage of the data flow hardware. Additionally, these functions are independent and asynchronous to each other.

PIO Sequences

PIO sequences are initiated by processor LOAD and STORE instructions.

DMA Sequences

DMA operations are initiated by the IOCC to gain access to system memory when the HRQ signal to the processor is activated.

Using the Hardware LEDs

The following procedure gives information on how the hardware uses system LEDs.

1. Enable the LEDs by accessing ROM. This is done with a read to ROM.
2. Read the R/C array. The last byte of the R/C array address equals the value of the LEDs, decimal numbers only. (For example, for the address X'811014', the LEDs would display the number 14. After the number has been displayed the LEDs can be blanked with a FF.)
3. Disable the LEDs by writing to the R/C array.

Memory Channel Interface

The memory channel interface connects the RT PC processor board with the system memory boards. This interface provides a multiplexed address bus that supports up to 16M-bytes of system memory, and a separate 40-bit data bus that contains 32 bits of data and 8 bits of error correcting code (ECC) check information.

The processor board contains address decode logic used to select the appropriate memory board based on the system memory address. This logic recognizes system memory board sizes of 512K-bytes, 2M-bytes, and 8M-bytes. Other memory board sizes are supported, with an unused address space between the memory board size and the next larger size recognized by the processor board. For example, with 1M-byte memory boards, addresses X'000000' to X'0FFFFF' are assigned to the first memory board, and addresses X'100000' to X'1FFFFF' are unused. If a second 1M-byte memory board is used, it is assigned addresses X'200000' to X'2FFFFF'.

Advanced Processor Board Description

The RT PC advanced processor board includes an IBM designed, custom logic proprietary 32-bit microprocessor, memory management unit (MMU), memory interface controller and two I/O interface chips. Features of these VLSI chips follow.

System Processor

The system processor is a single chip microprocessor that is mounted in a 175-pin package. The processor includes:

- A 32-bit processor on a single chip
  - Very Large Scale Integration (VLSI) custom CMOS design
  - A 32-bit address and data quantities
  - Single cycle execution of most register-to-register instructions
  - High memory bandwidth for CPU and I/O
- High performance virtual memory support
- Privileged and unprivileged states for system integrity
- A 32-bit timer facility
- Extensive error checking including:
  - Power On Reset (POR) microcode diagnostic
  - Programming errors (illegal instructions, invalid addresses)
  - Hardware errors (parity errors, timeout).
  - Overlapped Load/Store execution.

Detailed information on the system processor is provided in Section 11 and Appendix B.

Memory Management Unit (MMU)

The memory management unit implements a single level memory address translation architecture. The MMU contains all hardware required for address translation, translation lookaside buffer (TLB) management, and automatic reload on TLB miss.

The MMU is packaged in a 175-pin package. The MMU features include:

- Interfaces to the 32-bit microprocessor
- Performs virtual address translation
- VLSI custom CMOS design
  - A 40-bit virtual address
  - Supports 2K and 4K page sizes
  - Provides internal ECC logic for system memory
- Provides memory control to external system memory and IPL ROM
- Supports up to 16M-bytes of system memory
- Provides control of and access to reference and change bits.

Detailed information on the memory management unit is provided in Section 11.

Reference and Change (R/C) Arrays

The reference array and change arrays are 16K x 1 static RAMs. The R/C arrays store the reference and change bits associated with pages in memory. Access and use of the R/C arrays is described in "Reference And Change Bits" on page 11-111.

System Initial Program Load (IPL) ROM

The system IPL ROM contains basic hardware tests which are run everytime the hardware is turned on. Functions provided by the system IPL are described in "System IPL ROM" on page 7-4.

Memory Interface Controller (MIC)

The memory interface controller supports the following:

- Controls up to 16M bytes dynamic RAM
- Creates RAS's and CAS's - Refresh Address
- Controls R/C arrays
- Read and buffer ROM
- Latch and hold data for Read-Modify-Write (RMW) and three cycle writes
- Provides all Advanced Processor Board system clocks
- Support processor interface and control
- Drives RAS indicator LEDs
- 10K cell CMOS gate array.

I/O Interface Module

The I/O interface module includes:

- Enhanced DMA controller for Advanced Floating Point Accelerator
- Enhanced DMA controller for MC68881 chip
- Alternate DMA controller support for feature slot
- Enhanced diagnostics and wrapping capability
- Double word cache for DMA memory reads
  - Improves DMA read performance
  - Less PMUC impact
- Programmable interface options
- Status and error registers
- A 172 pin module
- Interface and protocol support for MC68881
- 10K cell CMOS gate array.

Floating Point Coprocessor (MC68881)

The floating point coprocessor supports:

- IEEE P754 Standards
- Full trig and transendental functions
- Virtual memory/machine operations
- A 32-bit data bus.

Advanced Processor Board Function Overview

The advanced processor board, as depicted in Figure 2-6 is divided into two major functional units and two interfaces. The system processor and the memory management unit are the two major components on the board. Four additional IBM custom designed modules are used on the processor board to create the external interface.

The two advanced processor board interface data flow blocks are shown in Figure 2-6.

"Advanced Processor Channel Interface" on page 2-23 describes the advanced processor channel interface.

"Advanced Processor Memory Channel Interface" on page 2-31 describes the advanced memory channel interface.

The advanced processor board interfaces are shown in Figure 2-7 on page 2-22.

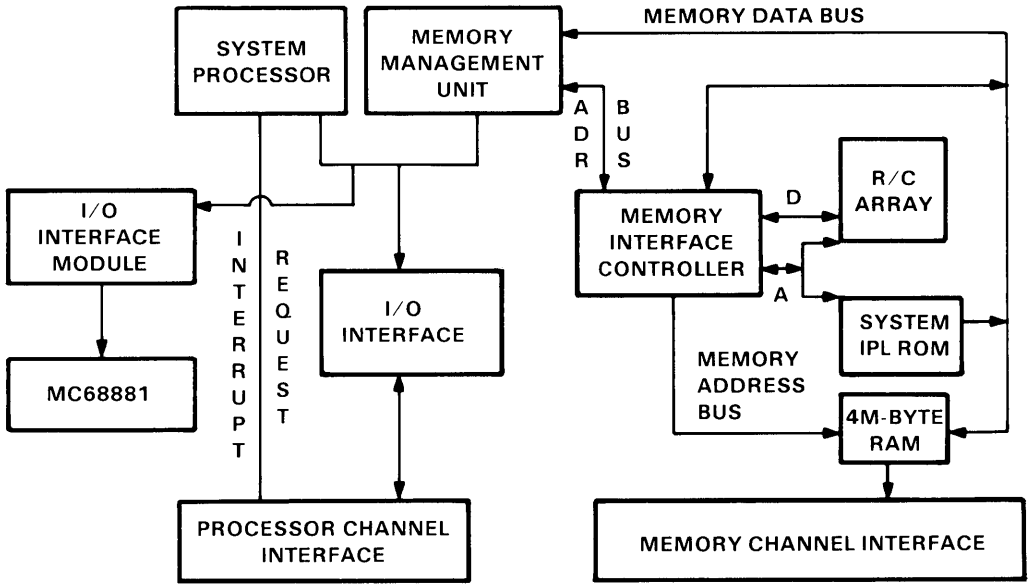


Figure 2-6. Advanced Processor Board Data Flow

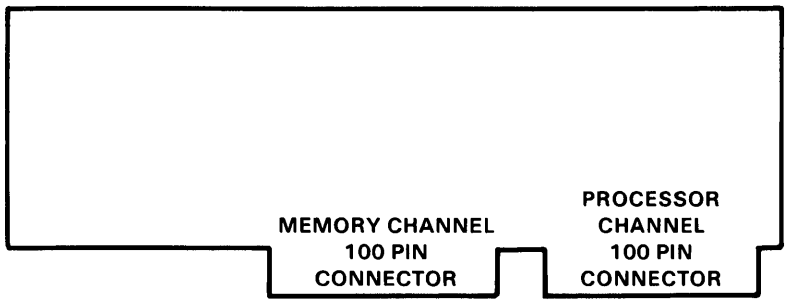


Figure 2-7. Advanced Processor Board Interfaces

Advanced Processor Channel Interface

The advanced processor channel interface connects the RT PC advanced processor board with the I/O subsystem and the Floating-Point Accelerator slot. It is a multiplexed 32-bit interface with appropriate signals to control programmed I/O (PIO) and direct memory access (DMA) sequences. Miscellaneous processor control lines such as interrupts and power on reset are also part of this channel section. This interface is driven by one of the I/O interface modules.

Common lines on the channel connect to both the IOCC and the Floating-Point Accelerator (FPA) board. Processor board input control lines include resistive pullups and may be driven with open collector drivers.

The advanced processor channel interface is illustrated in Figure 2-8 on page 2-24.

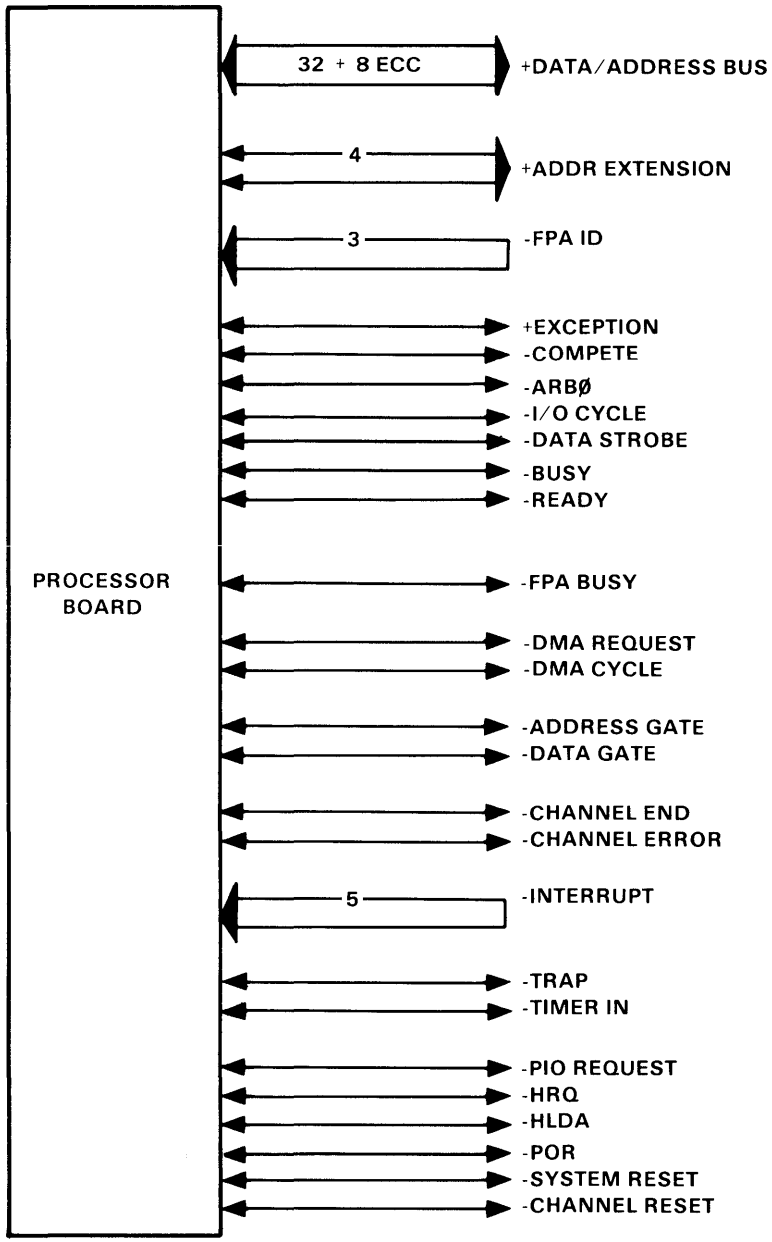


Figure 2-8. Advanced Processor Channel Interface

This illustration shows the connection of the advanced processor board, IOCC, and the Floating-Point Accelerator slot.

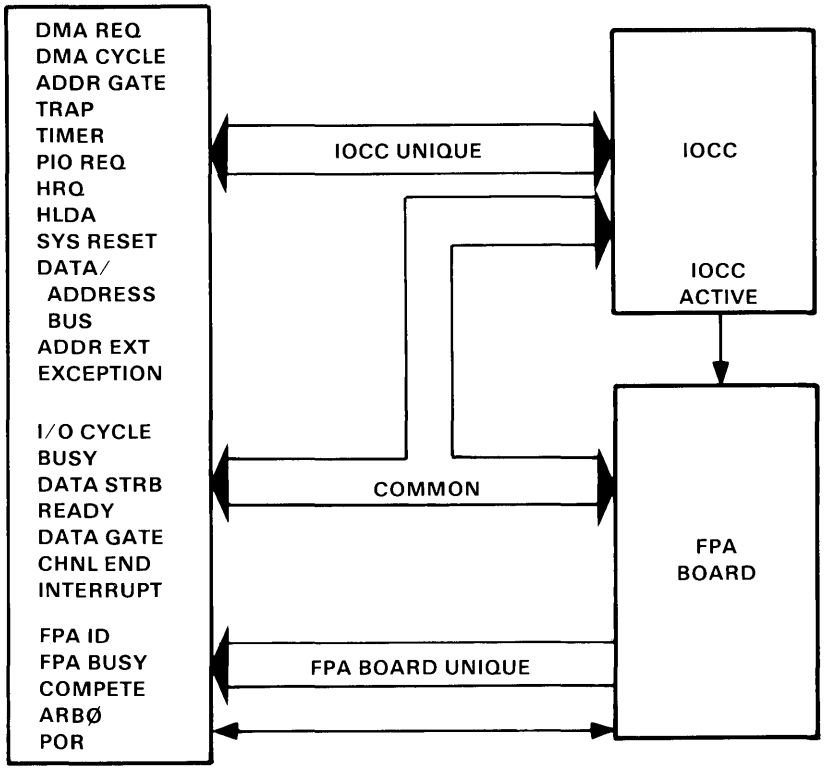


Figure 2-9. Advanced Processor Channel Interconnections

Signal Line Definitions

The processor channel interface consists of 74 signals and includes the following signal lines.

Data Address Bus

The 'data address bus' (DAB) is used to transmit address, data, and control between the processor and the I/O subsystem or Floating-Point Accelerator board.

Refer to Figure 2-10 for the data address bus format.

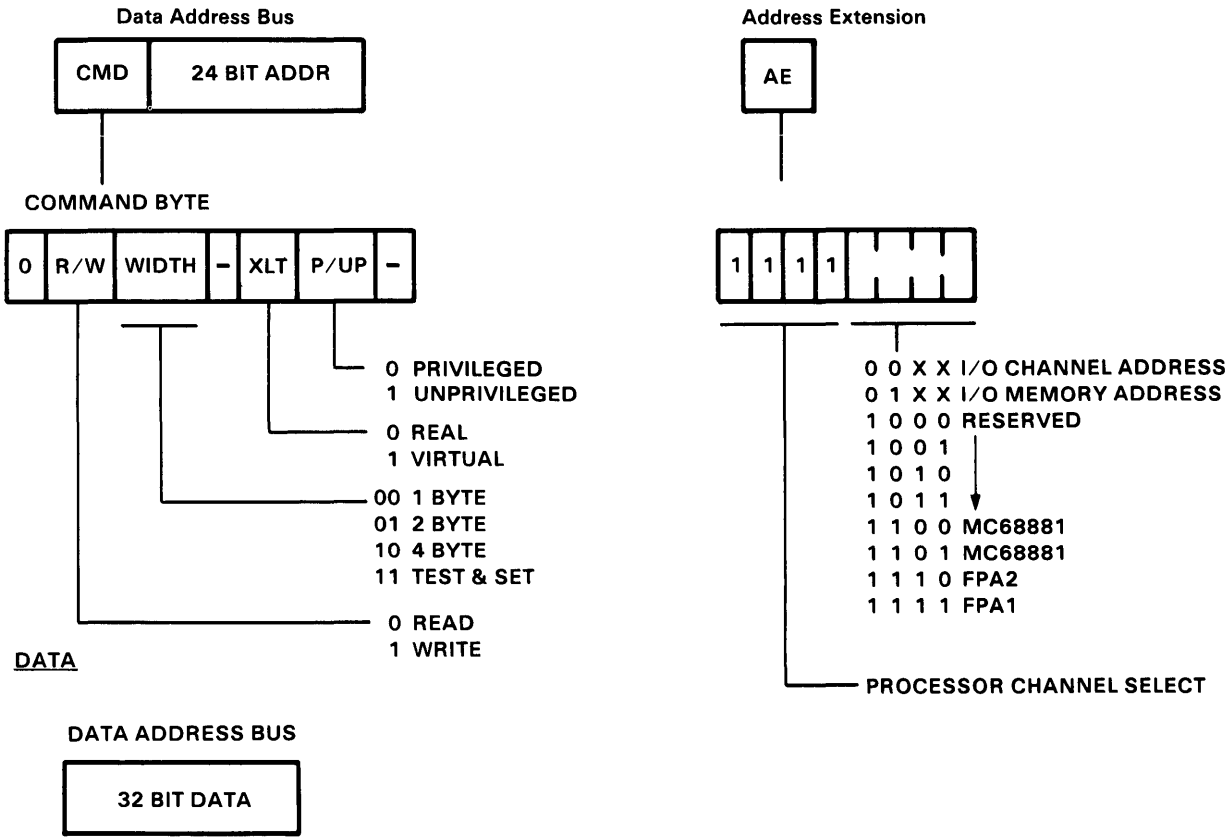


Figure 2-10. Data Address Bus Format

Address Extension

The 'address extension' provides subaddressing or control functions within the IOCC or Floating-Point Accelerator board devices.

+ Exception

'Exception' is an error response to PIO commands or DMA requests and indicates the operation cannot be executed or did not execute correctly.

-I/O Cycle

The 'I/O cycle' signal is activated whenever a PIO command is accepted.

-Data Strobe

'Data strobe' is used as a clock to signal that valid address or data is on the data bus.

-PIO Request

The 'PIO request' signal is activated
on queued PIO commands to I/O channel devices or I/O channel memory to signal the arbiter that the Coprocessor should be dearbitrated.

-Ready

The 'ready' signal indicates completion of channel operations.

-Busy

The 'busy' signal holds I/O cycle active. While active, no further (nondiagnostic) PIO commands are accepted from the processor.

-FPA Busy

The 'FPA busy' signal causes a (busy) rejection of PIO commands directed to the Floating-Point Accelerator slot.

-DMA Request

The 'DMA request' signal initiates a DMA memory cycle.

-DMA Cycle

The 'DMA cycle' is the response to a DMA REQUEST.

-Addr Gate

'Addr gate' controls tri-state drivers within the IOCC gating the I/O channel address onto the data bus.

-Data Gate

The 'data gate' signal controls tri-state drivers within the IOCC and Floating-Point Accelerator board for gating data or exception status onto the data bus.

-Channel Error

The 'channel error' signal is activated when a transmission error occurred from the IOCC or Floating-Point Accelerator board to the system.

-Interrupt

These 5 '-interrupt' signals are wired to the processor interrupt inputs for levels 0 - 4.

-Trap

The '-trap' signal is wired to the processor 'trap' input and may be used for functions such as early power-off warning. The 'trap' signal is edge sensitive.

-System Reset

The '-system reset' signal resets the system processor, and produces a 'channel reset' when the processor finishes its diagnostics. System memory is not disrupted by this signal.

-Channel Reset

The '-channel reset' signal is activated whenever a 'channel reset' is issued. This normally occurs on power up or after a processor error timeout.

-POR

'Power-on reset' is active during power on and is deactivated (100-500 ms) following all voltages becoming valid.

-HRQ

The '-HRQ' signal is activated by the IOCC arbiter to request usage of the IOCC data path and the I/O channel.

-HLDA

'HLDA' is the processor response to HRQ.

-Channel End

The '-channel end' signal is activated following presentation of ready. The low to high transition of this signal may be used as an indication of the end of the command.

-Timer In

The 'timer in' signal drives the processor timer.

-ARB0

'ARB0' is a combined bus request line and arbitration contention line that the feature device can use to obtain bus ownership, along with 'compete'.

-Compete

The 'compete' signal is driven by the current processor channel owner and indicates an arbitration period for determining the new channel owner.

-FPA ID

The 'FPA id' signal indicates which address combination the FPA responds to.

Arbitration

The various operations (PIO, DMA) performed or controlled by the IOCC are competitive in nature and require shared usage of the data flow hardware. Additionally, these functions are independent and asynchronous to each other.

PIO Sequences

PIO sequences are initiated by processor LOAD and STORE instructions.

DMA Sequences

DMA operations are initiated by the IOCC to gain access to system memory when the HRQ signal to the processor is activated.

Using the Hardware LEDs

The following procedure gives information on how the hardware uses system LEDs.

1. Enable the LEDs by accessing ROM. This is done with a read to ROM.
2. Read the R/C array. The last byte of the R/C array address equals the value of the LEDs, decimal numbers only. (For example, for the address X'811014', the LEDs would display the number 14. After the number has been displayed the LEDs can be blanked with a FF.)
3. Disable the LEDs by writing to the R/C array.

Advanced Processor Memory Channel Interface

The advanced processor memory channel interface connects the RT PC advanced processor board with the system memory boards. This interface provides a multiplexed address bus that supports up to 16M-bytes of system memory, and a separate 40-bit data bus that contains 32 bits of data and 8 bits of error correcting code (ECC) check information.

The advanced processor board contains address decode logic used to select the appropriate memory board based on the system memory address. This logic recognizes system memory board sizes of 512K-bytes, 1M-bytes, 2M-bytes, 4M-bytes and 8M-bytes. Other memory board sizes are supported, with an unused address space between the memory board size and the next larger size recognized by the processor board. For example, with 1M-byte memory boards, addresses X'000000' to X'0FFFFF' are assigned to the first memory board, and addresses X'100000' to X'1FFFFF' are unused. If a second 1M-byte memory board is used, it is assigned addresses X'200000' to X'2FFFFF'.

Processor Board Pin Assignments

The following tables contain the processor board slot pin assignments.

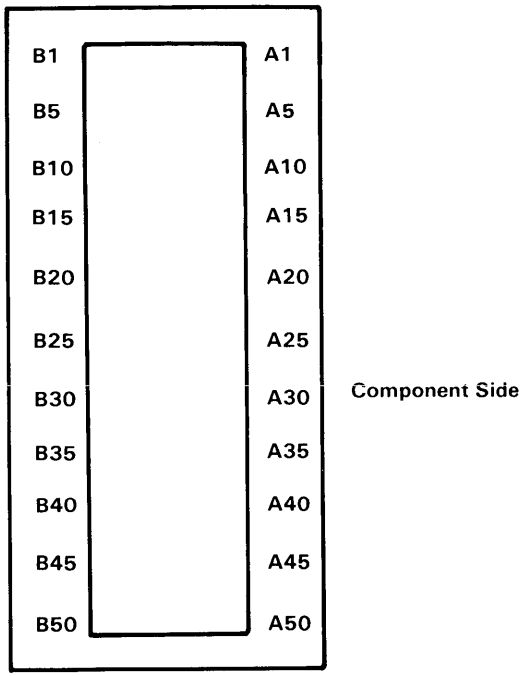


Figure 2-11. 100 Pin Processor Channel Interface Slot

I/O Pin	Signal Name	I/O
A 1	+ 5 Vdc	Power
A 2	+ 5 Vdc	Power
A 3	+ ADR/DATA 00	I/O
A 4	+ ADD/DATA 01	I/O
A 5	+ ADR/DATA 02	I/O
A 6	+ ADD/DATA 03	I/O
A 7	+ ADR/DATA 04	I/O
A 8	+ ADD/DATA 05	I/O
A 9	+ ADR/DATA 06	I/O
A 10	+ ADD/DATA 07	I/O
A 11	+ ADR/DATA 08	I/O
A 12	+ ADD/DATA 09	I/O
A 13	GND	Ground
A 14	+ ADR/DATA 10	I/O
A 15	+ ADD/DATA 11	I/O
A 16	+ ADR/DATA 12	I/O
A 17	+ ADD/DATA 13	I/O
A 18	+ ADR/DATA 14	I/O
A 19	+ ADD/DATA 15	I/O
A 20	+ ADR/DATA 16	I/O
A 21	+ ADD/DATA 17	I/O
A 22	+ ADR/DATA 18	I/O
A 23	+ ADD/DATA 19	I/O
A 24	GND	Ground
A 25	GND	Ground

Figure 2-12 (Part 1 of 2). Processor Channel Interface Slot (A-Side)

I/O Pin	Signal Name	I/O
A 26	+ ADD/DATA 20	I/O
A 27	+ ADR/DATA 21	I/O
A 28	+ ADD/DATA 22	I/O
A 29	+ ADR/DATA 23	I/O
A 30	+ ADD/DATA 24	I/O
A 31	+ ADR/DATA 25	I/O
A 32	+ ADD/DATA 26	I/O
A 33	+ ADR/DATA 27	I/O
A 34	+ ADD/DATA 28	I/O
A 35	+ ADR/DATA 29	I/O
A 36	+ ADR/DATA 30	I/O
A 37	GND	Ground
A 38	+ ADD/DATA 31	I/O
A 39	Reserved	—
A 40	-RSC HOLD	O
A 41	-IREQ 0	O
A 42	-IREQ 2	O
A 43	-IREQ 4	O
A 44	Reserved	—
A 45	+ Panel Ind 0	I
A 46	+ Panel Ind 2	I
A 47	+ Panel Ind 4	I
A 48	+ Panel Ind 6	I
A 49	+ 5 Vdc	Power
A 50	+ 5 Vdc	Power

Figure 2-12 (Part 2 of 2). Processor Channel Interface Slot (A-Side)

I/O Pin	Signal Name	I/O
B 1	GND	Ground
B 2	GND	Ground
B 3	+ ADR EXT 4	I
B 4	+ ADD EXT 5	I
B 5	+ ADR EXT 6	I
B 6	+ ADD EXT 7	I
B 7	-FPA ID 5	O
B 8	-Channel End	I
B 9	-System Reset	O
B 10	+ Parity 0	I/O
B 11	-Channel Reset	I
B 12	-POR	O
B 13	GND	Ground
B 14	-IOCC Busy	O
B 15	-Ready	O
B 16	-Channel Error	I
B 17	-I/O Cycle	I
B 18	+ Seg Sel	O
B 19	+ Parity 1	I/O
B 20	-Address Gate	I
B 21	-Data Gate	I
B 22	-FPA ID 6	O
B 23	-Data Strobe	I
B 24	+ 5 Vdc	Power
B 25	+ 5 Vdc	Power

Figure 2-13 (Part 1 of 2). Processor Channel Interface Slot (B-Side)

I/O Pin	Signal Name	I/O
B 26	-DMA Req	O
B 27	-HRQ	O
B 28	-HLDA	I
B 29	+ Parity 2	I/O
B 30	-DMA Cycle	I
B 31	-FPA ID 7	O
B 32	-FPA Busy	O
B 33	-Tag	I/O
B 34	-5 Vdc	Power
B 35	+ Exception	I/O
B 36	-Timer In	O
B 37	GND	Ground
B 38	+ Parity 3	I/O
B 39	-PIO Request	I
B 40	Reserved	—
B 41	-IREQ 1	O
B 42	-IREQ 3	O
B 43	Reserved	—
B 44	-I/O Trap	O
B 45	+ Panel Ind 1	I
B 46	+ Panel Ind 3	I
B 47	+ Panel Ind 5	I
B 48	+ Panel Ind 7	I
B 49	GND	Ground
B 50	GND	Ground

Figure 2-13 (Part 2 of 2). Processor Channel Interface Slot (B-Side)

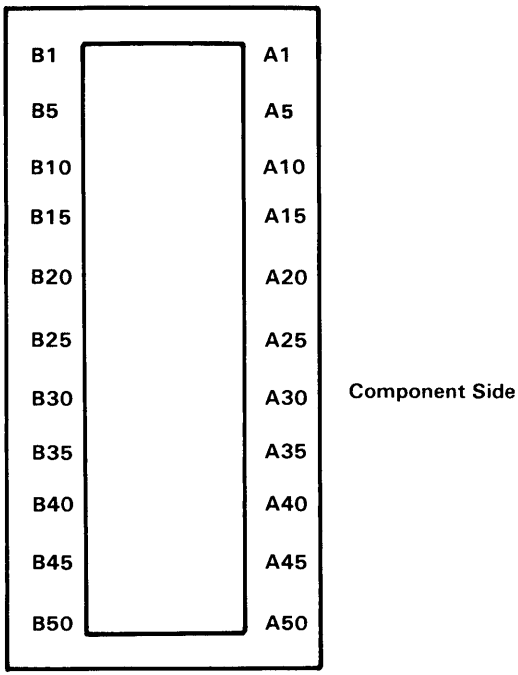


Figure 2-14. 100 Pin Memory Channel Interface Slot

I/O Pin	Signal Name	I/O
C 1	+5 Vdc	Power
C 2	+5 Vdc	Power
C 3	Cnfg 0 Slot C	O
C 4	Cnfg 1 Slot C	O
C 5	-Refresh Rate	O
C 6	Reserved	—
C 7	DATA 00	I/O
C 8	DATA 02	I/O
C 9	DATA 04	I/O
C 10	DATA 06	I/O
C 11	DATA 08	I/O
C 12	DATA 10	I/O
C 13	GND	Ground
C 14	Reserved	—
C 15	DATA 12	I/O
C 16	DATA 14	I/O
C 17	DATA 16	I/O
C 18	DATA 18	I/O
C 19	DATA 20	I/O
C 20	DATA 22	I/O
C 21	DATA 24	I/O
C 22	DATA 26	I/O
C 23	DATA 28	I/O
C 24	GND	Ground
C 25	GND	Ground

Figure 2-15 (Part 1 of 2). Memory Channel Interface Slot (C-Side)

I/O Pin	Signal Name	I/O
C 26	DATA 30	I/O
C 27	DATA 32	I/O
C 28	DATA 34	I/O
C 29	DATA 36	I/O
C 30	DATA 38	I/O
C 31	ADR A-9	I
C 32	ADR A-7	I
C 33	ADR A-5	I
C 34	ADR A-3	I
C 35	ADR A-1	I
C 36	ADR B-9	I
C 37	GND	Ground
C 38	ADR B-7	I
C 39	ADR B-5	I
C 40	ADR B-3	I
C 41	ADR B-1	I
C 42	RAS A Slot C	I
C 43	CAS A	I
C 44	WRT A	I
C 45	BEN A Slot C	I
C 46	RAS B Slot C	I
C 47	CAS B	I
C 48	WRT B	I
C 49	+ 5 Vdc	Power
C 50	+ 5 Vdc	Power

Figure 2-15 (Part 2 of 2). Memory Channel Interface Slot (C-Side)

I/O Pin	Signal Name	I/O
D 1	GND	Ground
D 2	GND	Ground
D 3	Cnfg 0 Slot D	O
D 4	Cnfg 1 Slot D	O
D 5	Cnfg 3	—
D 6	Reserved	—
D 7	DATA 01	I/O
D 8	DATA 03	I/O
D 9	DATA 05	I/O
D 10	DATA 07	I/O
D 11	DATA 09	I/O
D 12	DATA 11	I/O
D 13	GND	Ground
D 14	Bank Address Bit 0	I
D 15	DATA 13	I/O
D 16	DATA 15	I/O
D 17	DATA 17	I/O
D 18	DATA 19	I/O
D 19	DATA 21	I/O
D 20	DATA 23	I/O
D 21	DATA 25	I/O
D 22	DATA 27	I/O
D 23	DATA 29	I/O
D 24	+ 5 Vdc	Power
D 25	+ 5 Vdc	Power

Figure 2-16 (Part 1 of 2). Memory Channel Interface Slot (D-Side)

I/O Pin	Signal Name	I/O
D 26	DATA 31	I/O
D 27	DATA 33	I/O
D 28	DATA 35	I/O
D 29	DATA 37	I/O
D 30	DATA 39	I/O
D 31	ADR A-8	I
D 32	ADR A-6	I
D 33	ADR A-4	I
D 34	ADR A-2	I
D 35	ADR A-0	I
D 36	ADR B-8	I
D 37	GND	Ground
D 38	ADR B-6	I
D 39	ADR B-4	I
D 40	ADR B-2	I
D 41	ADR B-0	I
D 42	RAS A Slot D	I
D 43	Refresh	I
D 44	Bank Address Bit 1	I
D 45	BEN A Slot D	I
D 46	RAS B Slot D	I
D 47	BEN B Slot C	I
D 48	BEN B Slot D	I
D 49	GND	Ground
D 50	GND	Ground

Figure 2-16 (Part 2 of 2). Memory Channel Interface Slot (D-Side)

Section 3. System Memory Boards

CONTENTS

About this Section

This section contains information about the memory boards used by the IBM RT PC system. The memory board slots are defined and various memory configurations are listed. Also shown are the timing waveforms for the memory read, memory write and refresh cycles.

Memory

Two types of memory boards are supported; the first is the original memory board and the second is the fast memory board. The original memory board is utilized with the original processor board with processor cycle times of 170 nanoseconds. This memory board has an access time of 175 nanoseconds and a cycle of 290 nanoseconds. The fast memory board is utilized with the advanced processor board with processor cycle times of 100 nanoseconds. This fast memory board has an access time of 100 nanoseconds and a cycle time of 180 nanoseconds. Several aspects of the original memory board and the fast memory board are common to each other. The memory board descriptions in the following pages refers to both types of memory boards. Distinctions between the two memory boards is made by showing how the fast memory board differs from the original memory board.

The memory board contains two memory arrays. Each array is configured to be 40 bits wide and either 128K, 256K, 512K, or 1M address positions deep. These configurations yield board sizes of 1M-bytes, 2M-bytes, 4M-bytes, and 8M-bytes respectively.

The board interface consists of ten multiplexed address lines and four control lines for each array. There are 40 bidirectional data lines and 2 input control lines which are shared between the two arrays. It is the responsibility of the external control logic to ensure that there are no data bus usage conflicts between the two arrays.

Features

Two Way Interleaving	40 Bit Data Interface (32-bit data, 8-bit ECC)
High Density (1MB, 2MB, 4MB, 8MB)	
Single + 5V Supply	
Bandwidth	23.5MB/Sec.
Bandwidth (Fast)	40 MB/Sec.
Array Access Time	175ns
Array Access Time (Fast)	100ns
Availability	98%

Note: Bandwidth is based on processor cycle time.

Signal Definitions

RAS A	Row Address Strobe for Array A
RAS B	Row Address Strobe for Array B
CAS A	Column Address Strobe for Array A
CAS B	Column Address Strobe for Array B
WRT A	Write Enable and Bus Direction for Array A
WRT B	Write Enable and Bus Direction for Array B
BEN A	Bus Enable for Array A
BEN B	Bus Enable for Array B
ADR A (0-9)	Address Lines 0-9 for Array A (0 is LSB)
ADR B (0-9)	Address Lines 0-9 for Array B (0 is LSB)
DATA (00-39)	Data Lines 00-39 for Array A and Array B
CNFG (0-3)	Board Identification Jumpers
BAB 0	Bank Address Bit 0
BAB 1	Bank Address Bit 1
Refresh Rate	Refresh Rate Signal
Refresh	Refresh Input Signal

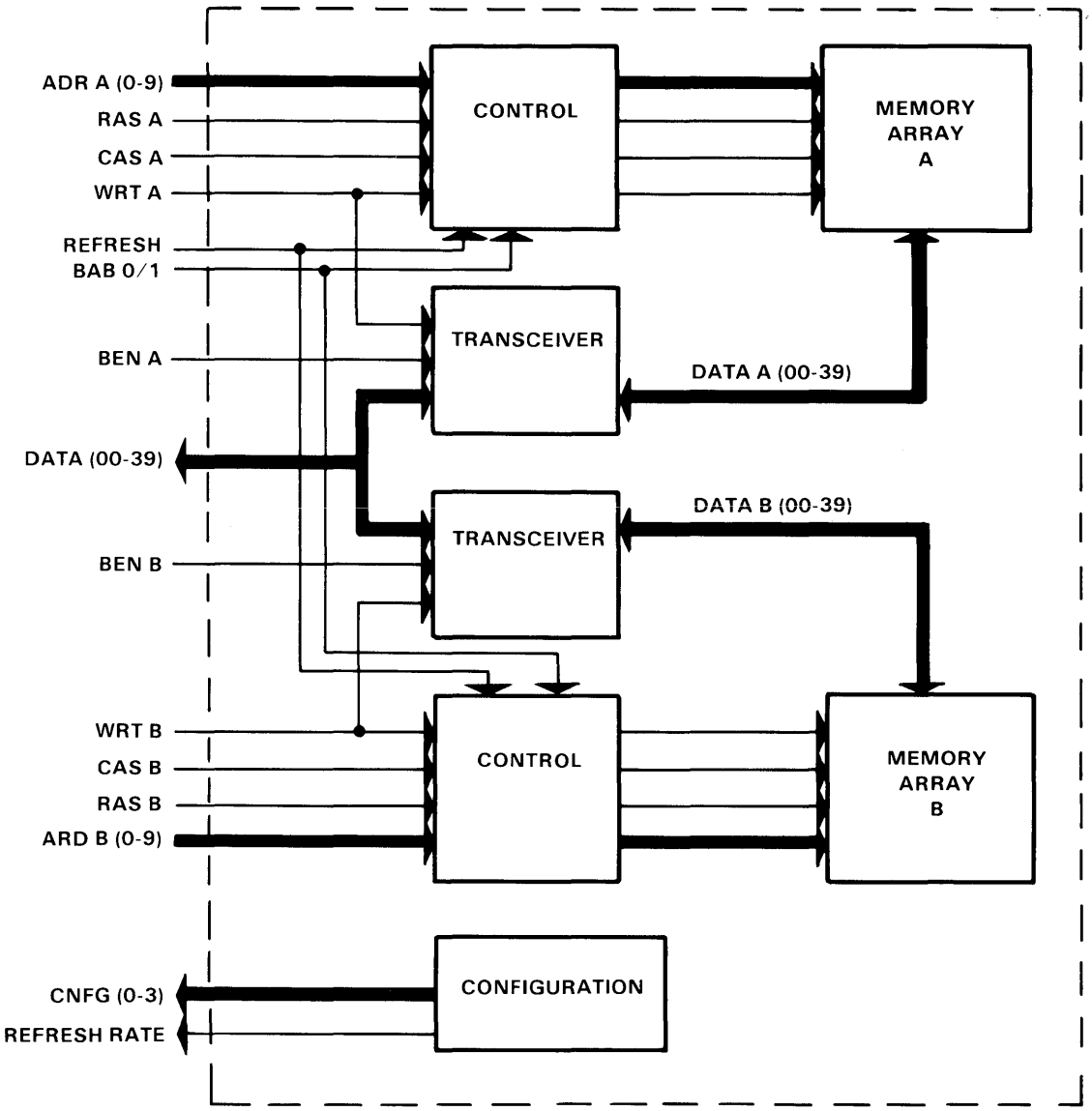


Figure 3-1. Memory Board Logical Dataflow

Configurations

The following tables list the various configurations for the memory boards.

Configuration Identification Table

CNFG3	CNFG2	CNFG1	CNFG0	Definition
1	X	1	X	Reserved
1	0	0	1	1MB - 175
1	1	0	1	2MB - 175
1	0	0	0	4MB - 175
1	1	0	0	8MB - 175
0	X	1	X	Reserved
0	0	0	1	1MB - 100
0	1	0	1	2MB - 100
0	0	0	0	4MB - 100
0	1	0	0	8MB - 100

Note:

0 = Tab pin grounded on the board

1 = Tab pin open (no connection)
Cnfg 3 selects fast (0) or original mode (1).

Bank Addressing Control for 1M-Byte Board

Each memory array can be subdivided into two banks which are 40 bits wide and 64K address positions deep.

The following bank addressing description is implemented independently for both memory arrays on the board. The eight least significant address bits (ADRA 0-7 and ADRB 0-7) are used to provide the 256 row and 256 column addresses (64K address space) to each bank of memory chips.

Bank address bit 0 and the Refresh (REF) signal are used to select between the two banks of each memory array. Bank address bit 0 and the REF input signal to the board are shared between the two arrays on the board.

The REF signal and Bank Address Bit 0 are routed through transparent latches. There is a separate set of latches for array A and array B and the latches are clocked at the beginning of a memory cycle with the rising edge of the RAS signal for that array. The latched values are then decoded and used to gate the incoming RAS signal to the proper memory bank.

The REF and BAB 0 signals have the same input timing requirements as the row address inputs. The logic required for bank addressing is implemented independently for each memory array on the board. The decode logic used for bank addressing is described by the following table.

RAS	REF	BAB 0	Bank Activated (By RAS Signal)
0	X	X	None
1	0	1	Bank 0 (1st bank of 64K x 4 chips)
1	0	0	Bank 1 (2nd bank of 64K x 4 chips)
1	1	X	Bank 0 and 1 (all banks Refresh)

Bank Addressing Control for 2M-Byte Board

Cards utilizing 256K x 1 DRAM modules use the nine multiplexed address bits (ADRA 0-8 and ADRB 0-8) to provide the 512 row and 512 column addresses (256K x 40 address space) to each array of memory chips. Refresh is handled by the external control logic. Therefore, the 'Refresh' (REF) signal and the Bank Address Bit (BAB 0) can be ignored.

Boards utilizing technologies other than 256K x 1 DRAM modules perform the addressing as described below. The following bank addressing description is implemented independently for both memory arrays on the board. The eight least significant address bits (ADRA 0-7 and ADRB 0-7) provide the 256 row and 256 column addresses (64K address space) to each bank of memory chips.

Each memory array can be subdivided into four banks which are 40 bits wide and 64K address positions deep.

To select between the four banks of each memory array, Bank Address Bit 0 (BAB 0), Row Address Bit 8 (ADRA 8 and ADRB 8) are used along with the Refresh signal. The BAB 0 and REF input signals to the board are shared between the two arrays on the board.

The REF signal, BAB 0, and ADR 8 are routed through transparent latches. There are separate sets of latches for Array A and Array B and the latches are clocked at the beginning of a memory cycle with the rising edge of the RAS signal for that array. The latched values are then decoded and used to gate the incoming RAS signal to the proper memory bank.

The REF and BAB 0 signals have the same input timing requirements as the row address inputs. The logic required for bank addressing is implemented independently for each memory array on the board. The decode logic used for bank addressing is described by the following table.

RAS	REF	ADR 8	BAB 0	Bank Activated (By RAS Signal)
0	X	X	X	None
1	0	1	1	Bank 0 (1st bank of 64K x 40)
1	0	1	0	Bank 1 (2nd bank of 64K x 40)
1	0	0	1	Bank 2 (3rd bank of 64K x 40)
1	0	0	0	Bank 3 (4th bank of 64K x 40)
1	1	X	X	Bank 0-3 (All banks Refresh)

Bank Addressing Control for 4M-Byte Board

Each memory array can be subdivided into two banks which are 40 bits wide and 256K address positions deep.

The following bank addressing description is implemented independently for both memory arrays on the board. The nine address bits (ADRA 0-8 and ADRB 0-8) provide the 512 row and 512 column addresses (256K address space) to each bank of memory chips.

To select between the two banks of each memory array, Bank Address Bit 1 (BAB 1), the Refresh (REF) signal, and optionally Address Bit 9 (ADRA 9 and ADRB 9) are used along with RAS. The BAB 1 and REF signals to the board are shared between the two arrays on the board.

The REF signal, BAB 1, and (optionally ADR 9) are routed through transparent latches. There are separate sets of latches for Array A and Array B and the latches are clocked at the beginning of a memory cycle with the rising edge of the RAS signal for that array. The latched values are then decoded and used to gate the incoming RAS signal to the proper memory bank.

The REF and BAB 1 signals have the same input timing requirements as the row address inputs. The logic required for bank addressing is implemented independently for each memory array on the board. The decode logic used for bank addressing is described by the following table.

RAS	REF	ADR 9	BAB 0	Bank Activated (By RAS Signal)
0	X	X	X	None
1	0	1	1	Bank 0 (1st bank of 256K x 40)
1	0	1	0	Bank 1 (2nd bank of 256K x 40)
1	0	0	1	None (or 1st bank of 256K x 40)
1	0	0	0	None (or 2nd bank of 256K x 40)
1	1	X	X	Bank 0 and 1 (All banks Refresh)

Note: The use of ADR 9 is optional for bank addressing control. Boards which use ADR 9 will not be activated when ADR 9 equals zero (0). Boards which treat ADR 9 as a 'don't care' will be activated by two distinct system address ranges (when ADR 9 = 0 and when ADR 9 = 1).

Bank Addressing Control for 8M-Byte Board

Each memory array can be subdivided into four banks which are 40 bits wide and 256K address positions deep.

Boards utilizing 1M x 1 DRAM modules use the ten multiplexed address bits (ADRA 0-9 and ADRB 0-9) to provide the 1024 row and 1024 column addresses (1M x 40 address space) to each array of memory chips. Refresh is handled by the external control logic. Therefore, the Refresh (REF) signal and the Bank Address Bit (BAB 1) can be ignored.

Boards utilizing technologies other than 1M x 1 DRAM modules perform the addressing as described below. The following bank addressing description is implemented independently for both memory arrays on the board. The nine least significant address bits (ADRA 0-8 and ADRB 0-8) are used to provide the 512 row and 512 column addresses (256K address space) to each bank of memory chips.

To select between the four banks of each memory array, Bank Address Bit 1 (BAB 1), Row Address Bit 9 (ADRA 9 and ADRB 9) are used along with the Refresh (REF) signal. The BAB 1 and REF signals to the board are shared between the two arrays on the board.

The REF signal, BAB 1, and ADR 9 are routed through transparent latches. There are separate sets of latches for Array A and Array B and the latches are clocked at the beginning of a memory cycle with the rising edge of the RAS signal for that array. The latched values are then decoded and used to gate the incoming RAS signal to the proper memory bank.

The REF and BAB 1 signals have the same input timing requirements as the row address inputs. The logic required for bank addressing is implemented independently for each memory array on the board. The decode logic used for bank addressing is described by the following table.

RAS	REF	ADR 9	BAB 0	Bank Activated (By RAS Signal)
0	X	X	X	None
1	0	1	1	Bank 0 (1st bank of 256K x 40)
1	0	1	0	Bank 1 (2nd bank of 256K x 40)
1	0	0	1	Bank 2 (3rd bank of 256K x 40)
1	0	0	0	Bank 3 (4th bank of 256K x 40)
1	1	X	X	Bank 0 - 3 (All banks Refresh)

Transceiver Control

Ben	WRT	Transceiver	Data Direction
1	X	Disabled	N.A.
0	0	Enabled	Memory array → board tab
0	1	Enabled	Board tab → memory array

Note: The transceiver control logic is implemented independently for each memory array on the board.

Absolute Maximum Ratings

Parameter	Limits	Unit	Notes
Operating temperature (TA)	+ 10 to + 70	°C	
Storage temperature (TS)	-55 to + 125	°C	1
Supply voltage (VCC)	-0.5 to + 6.0	V	1,2
Voltage on any pin except VCC	-1.0 to + 6.0	V	1,2

Notes:

1. These are stress ratings only. Functional operation at these limits is not implied.
2. Voltages relative to VSS.

DC Characteristics

		LIMITS				
Parameter	**Symbol**	**Min**	**Typical**	**Max**	**Unit**	**Test Condition**
Supply Voltages	VCC	4.75	5.0	5.25	V	
	VSS	0	0	0	V	All Voltages
High Input Voltage	VIH	2.4		5.5	V	Referenced
Low Input Voltage	VIL	-0.5		0.8	V	to VSS
High Output Voltage	VOH	2.4		VCC	V	
Low Output Voltage	VOL	0		0.4	V	
High Input Current	IIH			200	µA	
IIH for RAS Signal	IIHR			500	µA	
IIH for CAS, WRT, BAB	IIHC			400	µA	
Low Input Current	IIL			-2.0	mA	
IIL for RAS Signal	IILR			-10.0	mA	
IIL for CAS, WRT, BAB	IILC			-3.0	mA	
High Output Current	IOH			-2.0	mA	
Low Output Current	IOL			4.0	mA	
AVG Full Operating Supply Current both Arrays Active	ICC1			7200	mA	

		LIMITS				
AVG Standby Power Supply Current both Arrays Standby	ICC2			3000	mA	
Input Capacitance Data I/O Lines	CIND			25 (40)	pF	40 pF without Xceivers
Input Capacitance BEN Lines	CINB			25 (160)	pF	160 pF without Xceivers
Input Capacitance All Other Lines	CINC			25 (40)	pF	40 on fast memory

AC Characteristics (Original Memory)

		LIMITS			
Parameter	**Symbol**	**Min**	**Max**	**Unit**	**Notes**
Read or write cycle time	t_{RC}	290		ns	
Access time from RAS	t_{RAC}		175	ns	1, 2, 5
Access time from CAS	t_{CAC}		100	ns	1, 2, 5
Acc/turn on from BEN	t_{BAC}		40	ns	1
RAS precharge time	t_{RP}	120		ns	
RAS pulse width	t_{RAS}	165	10000	ns	
CAS Pulse width	t_{CAS}	90	10000	ns	
RAS to CAS delay time	t_{RCD}	55	75	ns	3
ROW address setup time	t_{ASR}	15		ns	
ROW address hold time	t_{RAH}	30		ns	
Column address setup time	t_{ASC}	15		ns	
Column address hold time	t_{CAH}	60		ns	
COL address hold time to RAS	t_{AR}	135		ns	
Read command setup time	t_{RCS}	25		ns	
Read command hold time	t_{RCH}	25		ns	
Write command setup time	t_{WCS}	40		ns	
Write command hold time	t_{WCH}	60		ns	
Write command hold to RAS	t_{WCR}	135		ns	
Write command pulse width	t_{WP}	100		ns	
Data in setup time	t_{DS}	10		ns	2, 6

		LIMITS			
Data in hold time	t_{DH}	60		ns	
Data in hold time to RAS	t_{DHR}	135		ns	
RAS hold time	t_{RSH}	90		ns	
CAS hold time	t_{CSH}	165		ns	
CAS to RAS precharge time	t_{CRP}	25		ns	
CAS precharge time	t_{CPN}	65		ns	
Data invalid from CAS	t_{OFF}	0	55	ns	
Buffer turn off from BEN	t_{OFFB}	0	40	ns	
Refresh period of A0 - A6	t_{REF1}		2	ms	4
Refresh period of A0 - A7	t_{REF2}		4	ms	4
Refresh period of A0 - A8	t_{REF3}		8	ms	4
Refresh period of A0 - A9	t_{REF4}		16	ms	4

Notes:

1. Measured with a load equivalent to 2 TTL loads and 100pF.
2. These parameters are valid when data transceivers are properly enabled.
3. The t_{RCD} (max) is for reference only. For $t_{RCD} < T_{RCD}$ (max) the access time (from RAS) = t_{RAC}. For $t_{RCD} > t_{RCD}$ (max) the access time (from RAS) = $t_{RCD} + t_{CAC}$.
4. The memory board meets either t_{REF1}, t_{REF2}, t_{REF3}, or t_{REF4}, the external controller is required to meet all specifications. For boards requiring a shorter refresh period, the refresh rate line on the board is grounded. This shortens the refresh period by one half.
5. For boards without TTL transceivers, these access times are reduced by 10ns. (t_{RAC} = 165ns and t_{CAC} = 90ns.)
6. For boards without TTL transceivers, this setup time is reduced by 10ns. (t_{DS} = 0ns)
7. An initial pause of 1ms is required after power up, followed by a minimum of eight initialization cycles to each chip on the board.

AC Characteristics (Fast Memory)

		LIMITS			
Parameter	**Symbol**	**Min**	**Max**	**Unit**	**Notes**
Read or write cycle time	t_{RC}	180		ns	
Access time from RAS	t_{RAC}		100	ns	1, 2, 5
Access time from CAS	t_{CAC}		60	ns	1, 2, 5
Access time from column address	t_{AA}		70	ns	
Acc/turn on from BEN	$_{BAC}$		25	ns	1
RAS precharge time	t_{RP}	80		ns	
RAS pulse width	t_{RAS}	90	10000	ns	
CAS Pulse width	t_{CAS}	55	10000	ns	
RAS to CAS delay time	t_{RCD}	30	45	ns	3
ROW address setup time	t_{ASR}	15		ns	
ROW address hold time	t_{RAH}	17		ns	
Column address setup time	t_{ASC}	10		ns	
Column address hold time	t_{CAH}	25		ns	
COL address hold time to RAS	t_{AR}	66		ns	
Read command setup time	t_{RCS}	10		ns	
Read command hold time	t_{RCH}	15		ns	
Write command setup time	t_{WCS}	35		ns	
Write command hold time	t_{WCH}	26		ns	
Write command hold to RAS	t_{WCR}	71		ns	
Write command pulse width	t_{WP}	30		ns	
Data in setup time	t_{DS}	10		ns	2, 6

		LIMITS			
Data in hold time	t_{DH}	30		ns	
Data in hold time to RAS	t_{DHR}	70		ns	
RAS hold time	t_{RSH}	50		ns	
CAS hold time	t_{CSH}	90		ns	
CAS to RAS precharge time	t_{CRP}	20		ns	
CAS precharge time	t_{CPN}	25		ns	
Data invalid from CAS	t_{OFF}	0	30	ns	
Buffer turn off from BEN	t_{OFFB}	0	20	ns	
Refresh period of A0 - A6	t_{REF1}		2	ms	4
Refresh period of A0 - A7	t_{REF2}		4	ms	4
Refresh period of A0 - A8	t_{REF3}		8	ms	4
Refresh period of A0 - A9	t_{REF4}		16	ms	4

Notes:

1. Measured with a load equivalent to 2 TTL loads and 100pF.
2. These parameters are valid when data transceivers are properly enabled.
3. The t_{RCD} (max) is for reference only. For $t_{RCD} < T_{RCD}$ (max) the access time (from RAS) = t_{RAC}. For $t_{RCD} > t_{RCD}$ (max) the access time (from RAS) = $t_{RCD} + t_{CAC}$.
4. The memory board meets either t_{REF1}, t_{REF2}, t_{REF3}, or t_{REF4}, the external controller is required to meet all specifications. For boards requiring a shorter refresh period, the refresh rate line on the board is grounded. This shortens the refresh period by one half.
5. For boards without TTL transceivers, these access times are reduced by 10ns. (t_{RAC} = 90ns and t_{CAC} = 50ns.)
6. For boards without TTL transceivers, this setup time is reduced by 10ns. (t_{DS} = 0ns)
7. An initial pause of 1ms is required after power up, followed by a minimum of eight initialization cycles to each chip on the board.

Timing Waveforms

All timings for Read, Write, and Refresh cycles are provided by an external memory controller. Read-Modify-Write cycles are not supported.

The address inputs (ADRA (0-9) and ADRB (0-9)) are numbered with bit 0 being the least significant. The Refresh (REF), Bank Address Bit 0 (BAB 0), or Bank Address Bit 1 (BAB 1) input lines for
bank addressing are valid during the row address valid time.

The external controller provides refresh for the lower 128 Row Addresses (ADRA (0-6) and ADRB (0-6)) every 2 milliseconds, refresh for the lower 256 Row Addresses (ADRA (0-7) and ADRB (0-7)) every 4 milliseconds, refresh for the lower 512 Row Addresses (ADRA (0-8) and ADRB (0-8)) every 8 milliseconds, and refresh for the lower 1024 Row Addresses (ADRA (0-9) and ADRB (0-9)) every 16 milliseconds. These refresh periods are reduced by a half for boards with the refresh rate signal grounded. RAS only refresh cycles are used to perform the refresh operation.

The timing marks in the following timing waveforms are referred to at two points. The lower point represents +0.8 volts and the upper point indicates +2.0 volts.

The timing waveforms in this section show only one array, but are required for both arrays. It is the responsibility of the external memory control logic to ensure that there are no data bus usage conflicts. The following timing waveforms are included in this section:

- Read cycle
- Write cycle
- Refresh cycle.

Read Cycle

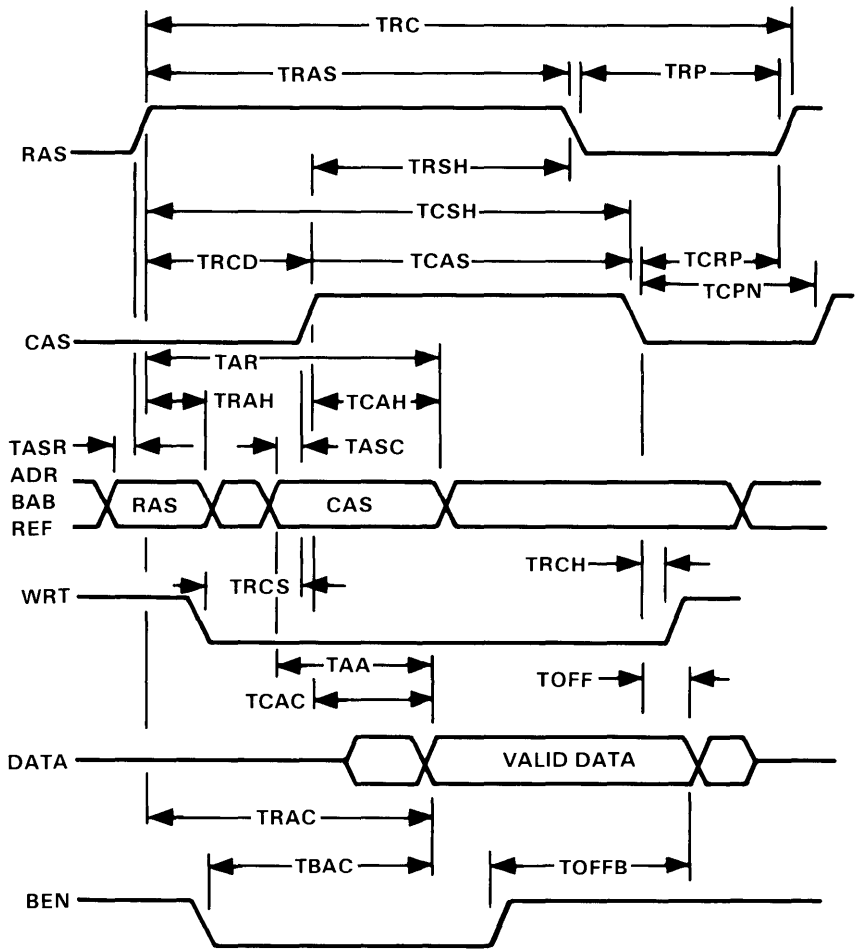


Figure 3-2. Memory Board Read Cycle Timing Waveform

Write Cycle

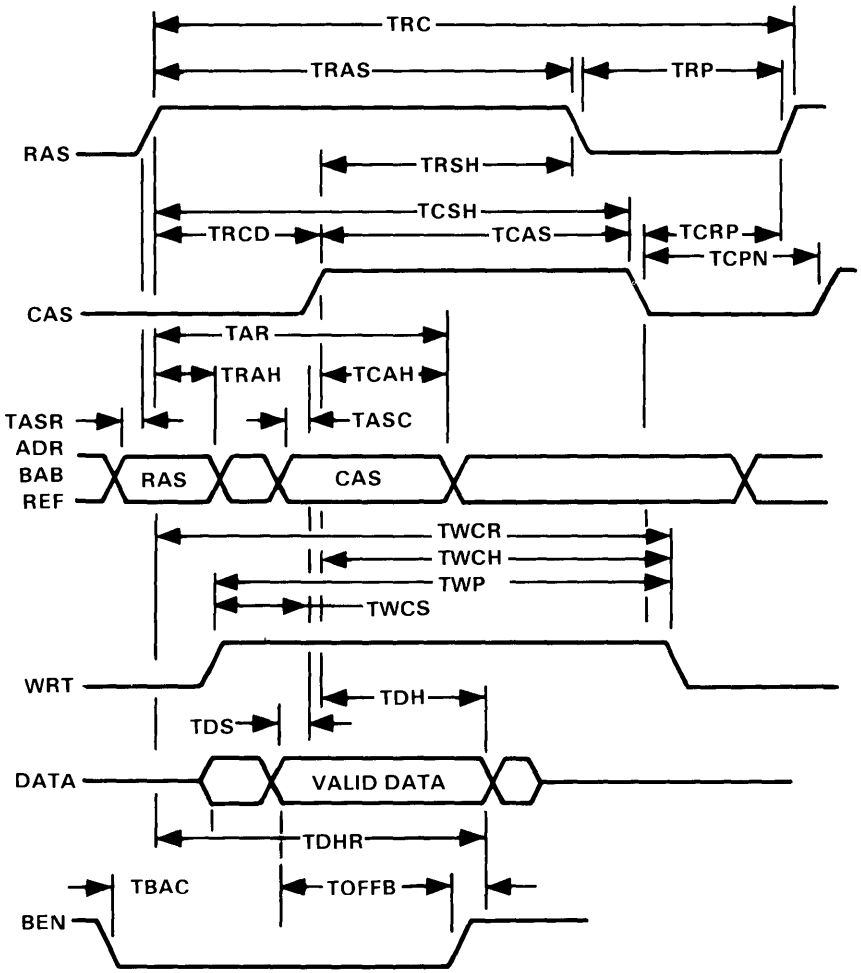


Figure 3-3. Memory Board Write Cycle Timing Waveform

Refresh Cycle

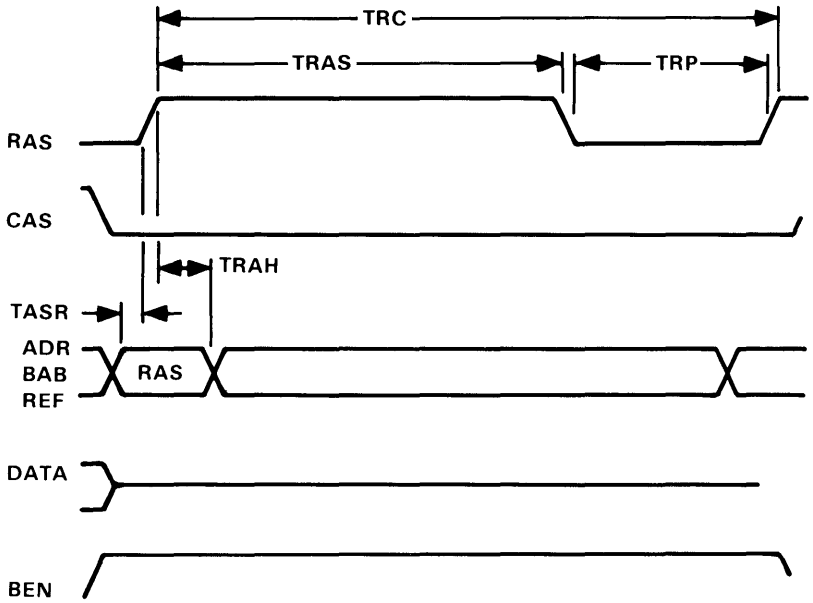


Figure 3-4. Memory Board Refresh Cycle Timing Waveform

Memory Board Pin Assignments

The following figures show the memory board pin assignments.

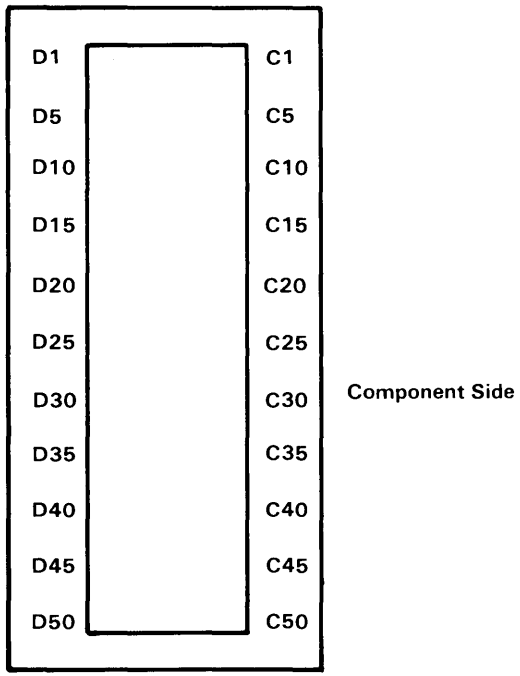


Figure 3-5. 100 Pin Memory Board Slot

I/O Pin	Signal Name	I/O
A 1	+ 5 Vdc	Power
A 2	GND	Ground
A 3	RAS A	I
A 4	CAS A	I
A 5	WRT A	I
A 6	BEN A	I
A 7	RAS B	I
A 8	CAS B	I
A 9	WRT B	I
A 10	BEN B	I
A 11	GND	Ground
A 12	ADR A-0	I
A 13	ADR A-2	I
A 14	ADR A-4	I
A 15	ADR A-6	I
A 16	ADR A-8	I
A 17	GND	Ground
A 18	+ 5 Vdc	Power
A 19	ADR B-0	I
A 20	ADR B-2	I
A 21	ADR B-4	I
A 22	ADR B-6	I
A 23	ADR B-8	I
A 24	Cnfg 3	O
A 25	GND	Ground

Figure 3-6 (Part 1 of 2). Memory Board Slot (A-Side)

I/O Pin	Signal Name	I/O
A 26	DATA 00	I/O
A 27	DATA 02	I/O
A 28	DATA 04	I/O
A 29	DATA 06	I/O
A 30	DATA 08	I/O
A 31	DATA 10	I/O
A 32	GND	Ground
A 33	+ 5 Vdc	Power
A 34	DATA 12	I/O
A 35	DATA 14	I/O
A 36	DATA 16	I/O
A 37	DATA 18	I/O
A 38	DATA 20	I/O
A 39	DATA 22	I/O
A 40	DATA 24	I/O
A 41	GND	Ground
A 42	DATA 26	I/O
A 43	DATA 28	I/O
A 44	DATA 30	I/O
A 45	DATA 32	I/O
A 46	DATA 34	I/O
A 47	DATA 36	I/O
A 48	DATA 38	I/O
A 49	GND	Ground
A 50	+ 5 Vdc	Power

Figure 3-6 (Part 2 of 2). Memory Board Slot (A-Side)

I/O Pin	Signal Name	I/O
B 1	+ 5 Vdc	Power
B 2	GND	Ground
B 3	Refresh Rate	O
B 4	Cnfg 0	O
B 5	Cnfg 1	O
B 6	Cnfg 2	O
B 7	Reserved	—
B 8	Refresh	I
B 9	Bank Address Bit 1	I
B 10	Bank Address Bit 0	I
B 11	GND	Ground
B 12	ADR A-1	I
B 13	ADR A-3	I
B 14	ADR A-5	I
B 15	ADR A-7	I
B 16	ADR A-9	I
B 17	GND	Ground
B 18	+ 5 Vdc	Power
B 19	ADR B-1	I
B 20	ADR B-3	I
B 21	ADR B-5	I
B 22	ADR B-7	I
B 23	ADR B-9	I
B 24	Reserved	—
B 25	GND	Ground

Figure 3-7 (Part 1 of 2). Memory Board Slot (B-Side).

I/O Pin	Signal Name	I/O
B 26	DATA 01	I/O
B 27	DATA 03	I/O
B 28	DATA 05	I/O
B 29	DATA 07	I/O
B 30	DATA 09	I/O
B 31	DATA 11	I/O
B 32	GND	Ground
B 33	+ 5 Vdc	Power
B 34	DATA 13	I/O
B 35	DATA 15	I/O
B 36	DATA 17	I/O
B 37	DATA 19	I/O
B 38	DATA 21	I/O
B 39	DATA 23	I/O
B 40	DATA 25	I/O
B 41	GND	Ground
B 42	DATA 27	I/O
B 43	DATA 29	I/O
B 44	DATA 31	I/O
B 45	DATA 33	I/O
B 46	DATA 35	I/O
B 47	DATA 37	I/O
B 48	DATA 39	I/O
B 49	GND	Ground
B 50	+ 5 Vdc	Power

Figure 3-7 (Part 2 of 2). Memory Board Slot (B-Side).

Section 4. Floating-Point Accelerator

CONTENTS

About this Section

This section describes the programmer's view of the Floating-Point Accelerator board. Many users will not require this level of information. Application programmers may require the Application Program Interface found in the AIX[1] Operating System Technical Reference. Floating point data representations and operation definitions are defined in the IEEE Standard 754 for Binary Floating-Point Arithmetic and are not detailed here.

[1] Trademark of International Business Machines Corporation

Floating-Point Accelerator

The RT PC Floating-Point Accelerator (FPA) is an optional system unit feature which can be installed in the IBM 6151 or IBM 6150. It utilizes a dedicated slot on the system board. Operands may be stored on the FPA and operations may be initiated by issuing processor load and store instructions. The FPA operates independently of the system unit so that the system unit and FPA operations can overlap. The board implements a subset of the IEEE 754 Standard for Binary Floating-Point Arithmetic (ANSI/IEEE STD 754-1985). Software is required to fully implement the standard. The board uses the National Semiconductor NS32081 floating point module, utilizing only the external memory functions of the part.

Block Diagram

Figure 4-1 on page 4-5 is a block diagram of the Floating-Point Accelerator board. When the system unit executes either a load or store instruction having any address in segment FF, the processor board routes the address and any associated data to the FPA over the 32 bit bus. The FPA decodes the command from the address.

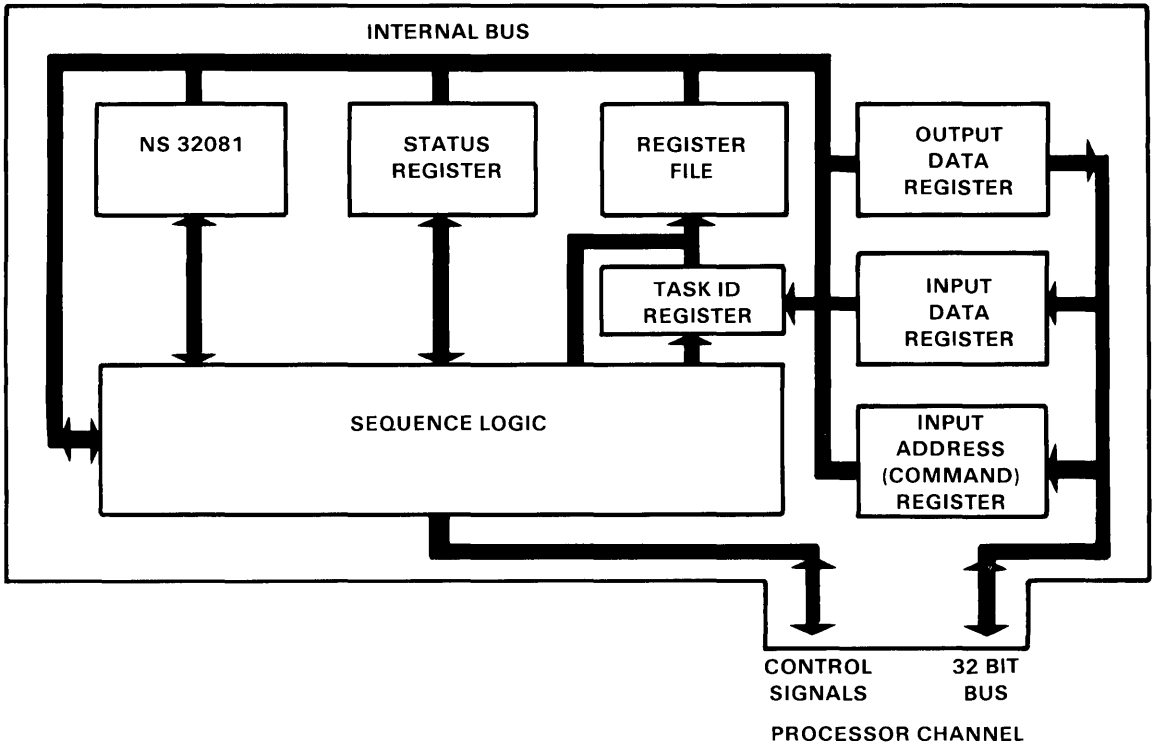


Figure 4-1. Floating-Point Accelerator Block Diagram

Data Formats

The FPA supports 2 floating-point data formats; the 32-bit single and the 64 bit double, as shown in Figure 4-2 on page 4-6 and Figure 4-3 on page 4-6.

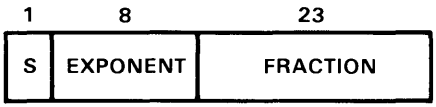


Figure 4-2. Single Precision Format

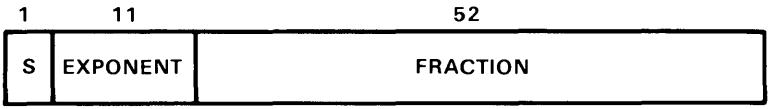


Figure 4-3. Double Precision Format

The Floating-Point Accelerator performs arithmetic in both formats, converts between formats, and converts to and from the system processor's twos complement binary format.

Register Sets

The register file contains 32 sets of 32 bit wide registers. Each set contains 16 registers. Long operands are stored with the high order 32 bits at an even register address and with the low 32 bits at the next higher odd register address. Context switching is achieved by issuing the Task Switch Unlock command which loads the Task ID register. Registers X'0 through D' are general purpose; register X'E' is the floating-point status register (FPSR) and register X'F' is the floating-point exception register (FPER). Figure 4-4 on page 4-7 shows the registers and how they are paired for long operands. Notice that there is a FPSR and a FPER in each of the 32 sets.

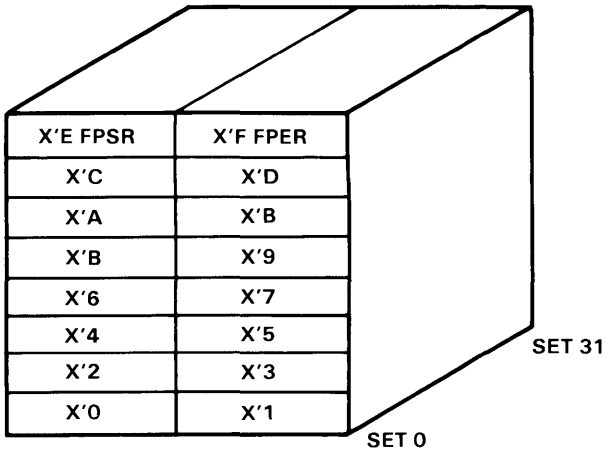


Figure 4-4. Floating Point Registers

Description

All transfers of commands and data between the system processor and the FPA are initiated by the system processor and accomplished by Store and Load instructions using memory mapped I/O. Memory addresses starting with X'FF' are decoded by the FPA as commands to it, and are ignored by other devices on the processor channel. The remaining 24 bits of the 32-bit memory address are used as the FPA command. If there is data associated with the command, it is transferred in the 32 bit data packet associated with the Store or Load instruction. If the command requires no data, the data packet is discarded. The system processor address and FPA command format is shown in Figure 4-5 on page 4-8.

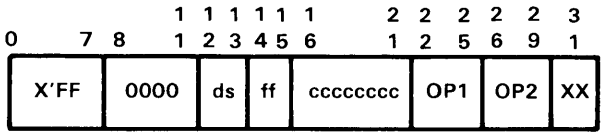


Figure 4-5. FPA Command Format

Bits 0 through 7 are the FPA address and are fixed at X'FF'. Bits 8 through 11 are reserved and must be 0. Bits 12 and 13 indicate if the data packet is to be used and whether it is to go into the register indicated by OP1 or OP2. Bits 14 and 15 are format bits which can be regarded as an extension of the command field, bits 16 through 21. Bits 30 and 31 are reserved bits. Notice that the format bits assume values 00, 01, and 10 only.

Data Source Field Definitions

Bits 12-13	Function
00	Non immediate data
01	Immediate data to OP2 address
10	Immediate data to OP1 address
11	Reserved

Format Field Definitions

Bits 14-15	Function
00	Format 0 - FPU type 9
01	Format 1 - FPU type 11
10	Format 2 - Non FPU
11	Reserved

Bits 0 through 3 do not actually reach the FPA and bits 4 through 7 become an address extension. Bits 0 through 7 however, are replaced in the 32 bit instruction which reaches the FPA with the 8-bit processor channel control word providing FPA control and exception reporting. The two operand fields, OP1 and OP2, specify the FPA source and destination addresses, respectively.

Most floating-point instructions are unprivileged instructions. Task Switch/Unlock, Lock, and Read Status/Clear Exception are valid only in privileged state.

Normally, the FPA should be used in the virtual mode of operation in which program checks (exception conditions) are reported on both Load and Store instructions. FPA exceptions are caused by parity errors, illegal commands or operands, or various data situations, such as overflow. The FPA can be used in the real mode of operation if certain rigorous programming restrictions are followed (see "FPA Programming Considerations" on page 4-40).

FPA Status Register (FPSR)

The FPA status register, shown in Figure 4-6 is a hardware register used to control the FPA operation, (such as rounding mode) and to identify various exception conditions. The last state of the status register is saved in register X'E' of the active register set by a Read Status Register instruction.

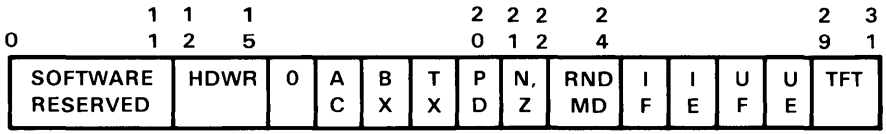


Figure 4-6. Floating Point Status Registers

Bits 0 through 15 and 20 through 31 may be written or read by status register commands. Bits 21, 22, 25, 27, and 29 through 31, however, may change during the operations of the FPA. Bits 16 through 19 cannot be written by the programmer.

Status Register Definitions

Bits	Function
0-11	Useable by software; bits 0 through 9 utilized by IBM software.
12-15	Reserved for hardware uses.
16	Permanent 0.
17	1 = Aborted Command Board was reset while busy; set by hardware; reset only by RDSCX command. Does not cause program check. See command list for RDSCX.
18	Bus Exception 1 = Program check due to bus parity error on last FPA command. Set by hardware; reset by RDSCX. Nonmaskable.

19 Task Exception

1 = Task Exception; set by hardware, indicates that an FPA exception is causing a program check, reset by RDSCX. Set on last command if a type 7 exception is detected, set on second to last command if bit 20 is 0 and an enabled or ungatable exception type 1-6 occurs. (See bits 26 through 28 and 29 through 31.)

20 Program Check Disable

1 = Program checks disabled on exception types 1-6. (See bits 29 through 31.) Normally set to 0.

21-22 N, Z:

These bits hold the results of the last compare operation:

00 = 0P1<0P2
01 = 0P1 = 0P2
10 = 0P1>0P2
11 = (Software may reserve this state for unordered.)

23-24 Rounding Mode Control:

00 = Round (nearest)
01 = Truncate (towards zero)
10 = Ceiling (towards + ∞)
11 = Floor (towards - ∞)

25 Inexact Flag:

1 = Inexact result occurred since this bit was written to zero.
0 = No inexact result since bit was written to zero.

26 Inexact Enable

1 = No result produced on inexact result. Also program check occurs if Bit 20 = 0.
0 = No program check occurs, results rounded per bits 23-24.

27 Underflow Flag

1 = Underflow occurred since this bit was written to zero.
0 = No underflow since this bit was written to zero.

28 Underflow Enable

1 = No result produced on UF; and program check occurs if 111120 = 0.
0 = No program check occurs and result set to 0 on UF.

29-31 Exception Type

These bits are changed on each command performed except WTFR, RDFR, RDSTR, RDSCX regardless of bits 20, 26 or 28.

Type	29	30	31	
0	0	0	0	No Exception
1	0	0	1	Underflow – See above
2	0	1	0	Overflow – result register unchanged
3	0	1	1	Divide by zero – result register unchanged
4	1	0	0	Reserved
5	1	0	1	Operand Error – result register unchanged
6	1	1	0	Inexact result – see above
7	1	1	1	Illegal Command – reset only with RDSCX; Nonmaskable

Note: Conversion of very large floating-point numbers to word may result in an exception. If this occurs, it results in exception type 2 rather than 5.

FPA Exception Register (FPER)

When an exception is reported, Register X'F' of the selected register group will contain the following information about the last executed command. The first 8 bits are taken from the channel control word and the remaining bits are taken from the command.

Bit	Meaning
0	0
1	0 = Load; 1 = Store
2	1
3	0
4	1 = Storage Protect Enabled (not used by FPA)
5	0 = Real Mode; 1 = Virtual Mode
6	0 = Privileged State; 1 = Nonprivileged (Normal State)
7	Reserved
8-31	Bits 8-31 of the command.

Note: A 4 byte (32 bit) operation is selected when bits 2 and 3 equal 10.

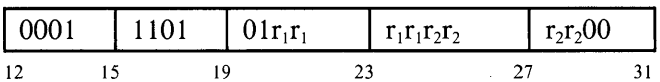
Command Encoding (Effective Address of Load or Store)

r_1, r_2	Binary weighted bits specifying one of the 16 active short registers or 8 active long registers pairs as operand 1 or 2.
R_1,R_2	32-bit register for operand 1, operand 2, or result, specified by r_1,r_2.
(R_1), (R_2)	Contents of register R_1,R_2.
I	32 bits of immediate data associated with a store type command.
RP_1,RP_2	64-bit register pair for operand 1, operand 2, or result, specified by r_1,r_2.
(RP_1), (RP_2)	Contents of register pair RP_1,RP_2.
Su	Indicates that exception will result if system not in supervisory state.
Ld	Load type instruction.
St	Store instruction.
Boxed area	Bits 12-31 of the address of the load or store instruction. Bits 0-11 are always X'FF0'.
→	Replaces the content of.

Floating-Point Commands

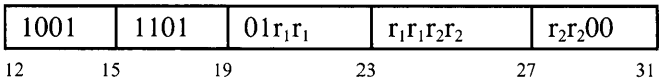
The floating-point commands are described and listed in alphabetical order. Some other commands, not documented here, are legal but redundant, and are not supported.

Absolute Short (ABSS)



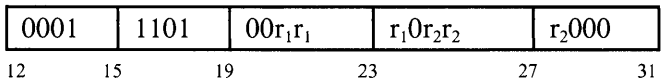
St | (R_1) | → R_2

Absolute Immediate Short (ABSIS)



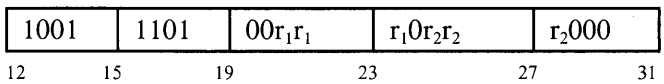
St $I \rightarrow R_1$
$|(R_1)| \rightarrow R_2$
(R_1) preserved

Absolute Long (ABSL)



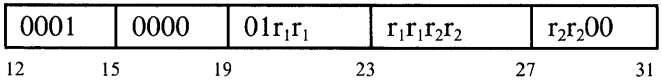
St $|(RP_1)| \rightarrow RP_2$

Absolute Immediate Long (ABSIL)



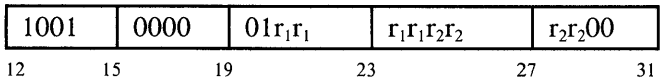
St $I \rightarrow R_1$ (32 bits - high half of 64)
$|(RP_1)| \rightarrow RP_2$
(RP_1) preserved

Add Short (ADDS)



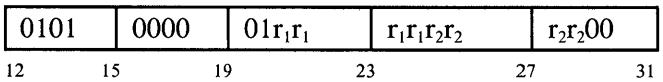
St $(R_1) + (R_2) \rightarrow R_2$

Add Immediate Short (ADDIS)



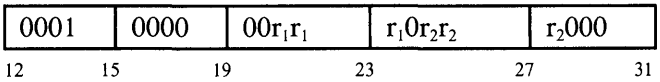
St $I \rightarrow R_1$
$(R_1) + (R_2) \rightarrow R_2$

Add Short Immediate (ADDSI)



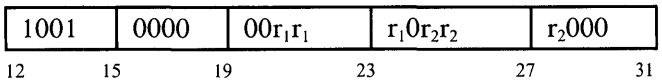
$I \rightarrow R_2$
St $(R_1) + (R_2) \rightarrow R_2$

Add Long (ADDL)



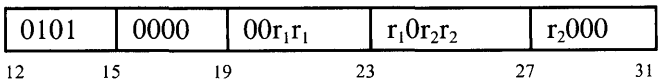
St $(RP_1) + (RP_2) \rightarrow RP_2$

Add Immediate Long (ADDIL)



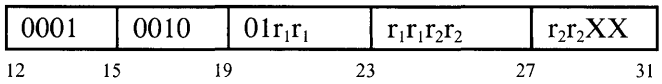
St $I \rightarrow R_1$ (32 bits – high half of 64)
$(RP_1) + (RP_2) \rightarrow RP_2$

Add Long Immediate (ADDLI)



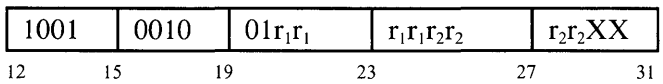
St $I \rightarrow R_2$ (32 bits – high half of 64)
$(RP_1) + (RP_2) \rightarrow RP_2$

Compare Short (COMS)



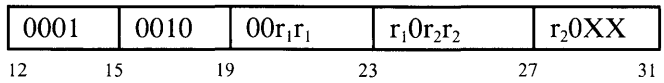
St (R_1) compared to (R_2)
Status register N, Z set (see status register bits 21-22)

Compare Immediate Short (COMIS)



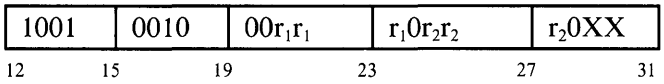
St $I \rightarrow R_1$
(R_1) compared to (R_2)
Status register N, Z set (see status register bits 21-22)

Compare Long (COML)



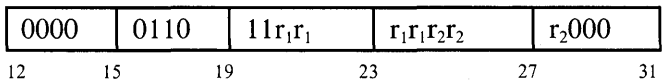
St (RP_1) compared to (RP_2)
Status register N, Z set (see status register bits 21-22)

Compare Immediate Long (COMIL)



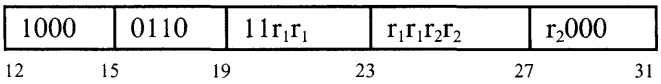
St $I \rightarrow R_1$ (32 bits – high half of 64)
(RP_1) compared to (RP_2)
Status register N, Z set (see status register bits 21-22)

Convert Short To Long (CSL)



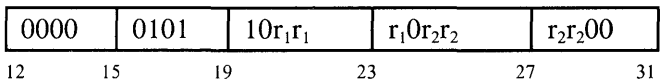
St (R_1) (32 bits) converted to long format (64 Bits)
(R_1) preserved
Result $\rightarrow RP_2$

Convert Immediate Short To Long (CISL)



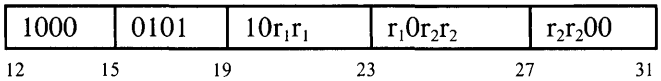
St $I \rightarrow R_1$ (32 bits)
(R_1) converted to long format (64 bits)
(R_1) preserved
Result $\rightarrow RP_2$

Convert Long To Short (CLS)



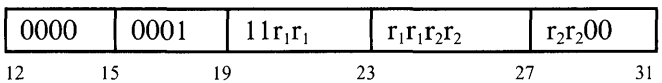
St (RP_1) converted to short format (32 bits)
(RP_1) preserved
Result $\rightarrow R_2$

Convert Immediate Long To Short (CILS)



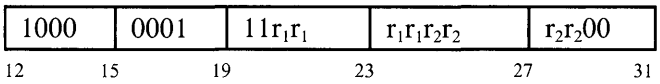
St $I \rightarrow R_1$ (32 bits – high half of 64)
(RP_1) converted to short format
(RP_1) preserved
Result $\rightarrow R_2$

Convert Word To Short (CWS)



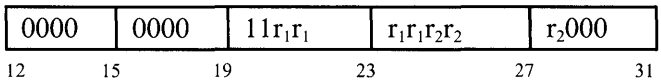
St (R_1) converted to short float format
(R_1) preserved
Result $\rightarrow R_2$

Convert Immediate Word To Short (CIWS)



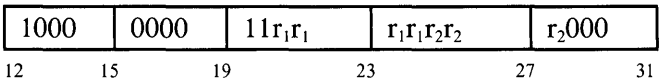
St I → R_1 (32 bit integer)
(R_1) converted to short float format
(R_1) preserved
Result → R_2

Convert Word To Long (CWL)



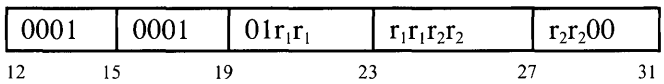
St (R_1) converted to long float format
(R_1) preserved
Result → RP_2

Convert Immediate Word To Long (CIWL)



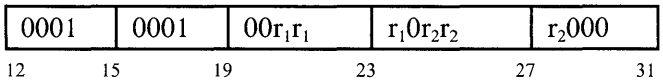
St $I \rightarrow R_1$ (32 Bit Integer)
(R_1) converted to long float format
(R_1) preserved
Result $\rightarrow RP_2$

Copy Short (COPS)



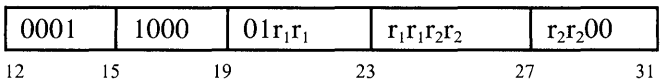
St $(R_1) \rightarrow R_2$

Copy Long (COPL)



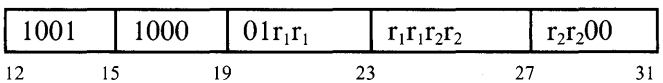
St $(RP_1) \rightarrow RP_2$

Divide Short (DIVS)



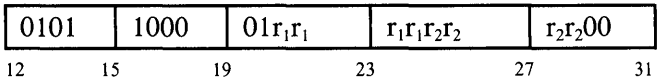
St $(R_2) \div (R_1) \rightarrow R_2$

Divide Immediate Short (DIVIS)



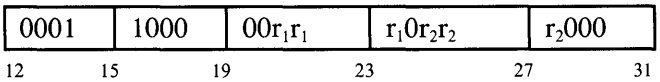
St $I \rightarrow R_1$
$(R_2) \div (R_1) \rightarrow R_2$

Divide Short Immediate (DIVSI)



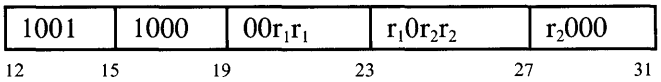
St $I \rightarrow R_2$
$(R_2) \div (R_1) \rightarrow R_2$

Divide Long (DIVL)



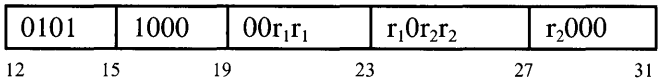
St $(RP_2) \div (RP_1) \rightarrow RP_2$

Divide Immediate Long (DIVIL)



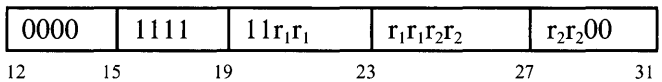
St $I \rightarrow R_1$ (32 bits — high half of 64)
$(RP_2) \div (RP_1) \rightarrow RP_2$

Divide Long Immediate (DIVLI)



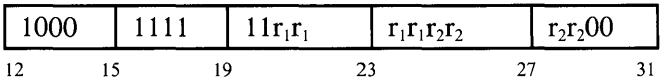
St $I \rightarrow R_2$ (32 bits — high half of 64)
$(RP_2) \div (RP_1) \rightarrow RP_2$

Floor Short To Word (FSW)



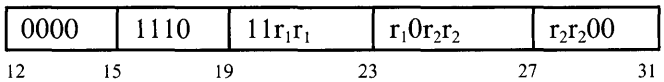
St (R_1) converted to integer and floored
(R_1) preserved
Result $\rightarrow R_2$

Floor Immediate Short To Word (FISW)



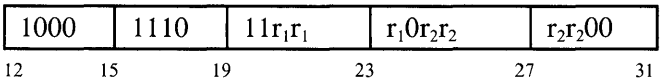
St $I \rightarrow R_1$
(R_1) converted to integer and floored to 32 bits
(R_1) preserved
Result $\rightarrow R_2$

Floor Long To Word (FLW)



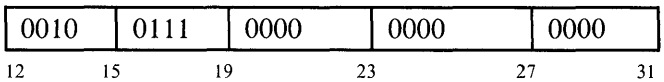
St (RP_1) converted to integer and floored to 32 bits
(RP_1) preserved
Result $\rightarrow R_2$

Floor Immediate Long To Word (FILW)



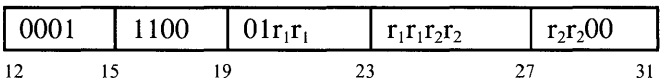
St $I \rightarrow R_1$ (32 bits - high half of 64)
(RP_1) converted to integer and floored to 32 bits
(RP_1) preserved
Result $\rightarrow R_2$

Lock Floating Point (LOCKFP) - Privileged Instruction



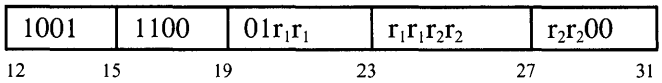
St, Su Locks out all instructions except TSKSWU, RDSCX, LOCKFP (Type 7 exception generated upon receiving a locked out instruction).

Multiply Short (MULS)



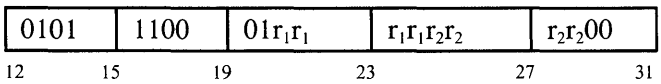
St $(R_1) \times (R_2) \rightarrow R_2$

Multiply Immediate Short (MULIS)



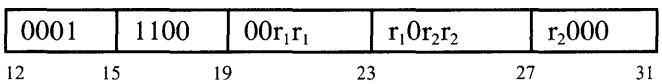
St $I \rightarrow R_I$
$(R_1) \times (R_2) \rightarrow R_2$

Multiply Short Immediate (MULSI)



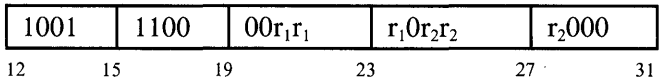
St $I \rightarrow R_2$
$(R_2) \times (R_1) \rightarrow R_2$

Multiply Long (MULL)



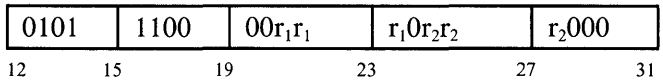
St $(RP_1) \times (RP_2) \rightarrow RP_2$

Multiply Immediate Long (MULIL)



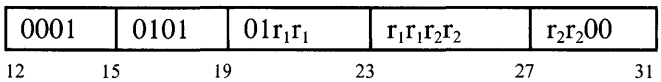
St $I \rightarrow R_1$ (32 bits – high half of 64)
$(RP_1) \times (RP_2) \rightarrow RP_2$

Multiply Long Immediate (MULLI)



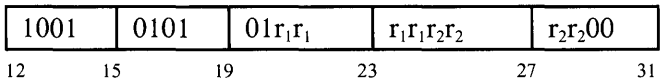
St $I \rightarrow R_2$ (32 Bits – high half of 64)
$(RP_1) \times (RP_2) \rightarrow RP_2$

Negate Short (NEGS)



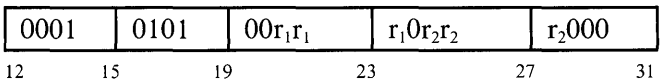
St $(R_1) \times (-1) \rightarrow R_2$

Negate Immediate Short (NEGIS)



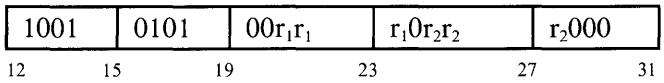
St $I \rightarrow R_1$
$(R_1) \times (-1) \rightarrow R_2$
(R_1) preserved

Negate Long (NEGL)



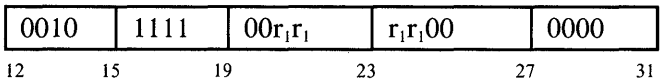
St $(RP_1) \times (-1) \rightarrow RP_2$

Negate Immediate Long (NEGIL)



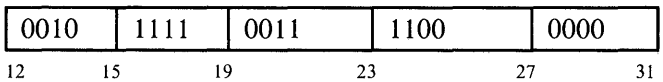
St $I \rightarrow R_1$ (32 bits - high half of 64)
$(RP_1) \times (-1) \rightarrow RP_2$
(RP_1) preserved

Read Float Register (RDFR)



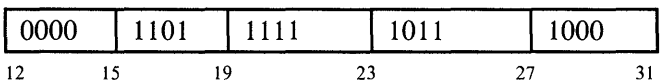
Ld $(R_1) \rightarrow$ CPU (valid R_1 = X'0-D')
(for 64 bits perform second RDFR at $R_1 + 1$)

Read Exception Register (RDER)



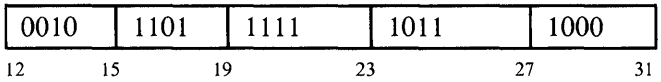
Ld (Reg X'F') (FPER) → CPU (see FPER description)

Read Status Register (RDSTR)



Ld Hardware status → Reg X'E' then (Reg X'E') → CPU

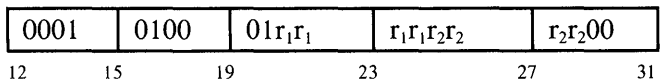
Read Status Clear Exception (RDSCX) - Privileged Instruction



Ld, Su Hardware status → CPU → REG X'E' of active register set

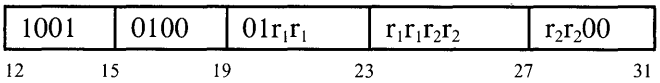
- Program check conditions and associated status bits reset
- Only used at initialization or following an exception.

Round Short To Word (RSW)



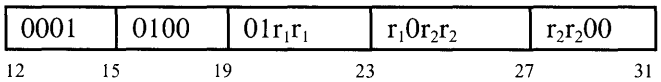
St (R_1) converted to integer and rounded to 32 bits
(R_1) preserved
Result → R_2

Round Immediate Short To Word (RISW)



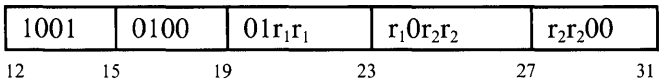
St $I \rightarrow R_1$
(R_1) converted to integer and rounded to 32 bits
(R_1) preserved
Result $\rightarrow R_2$

Round Long To Word (RLW)



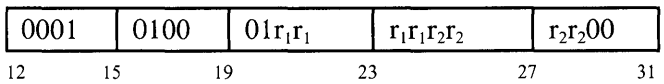
St (RP_1) converted to integer and rounded to 32 bits
(RP_1) preserved
Result $\rightarrow R_2$

Round Immediate Long To Word (RILW)



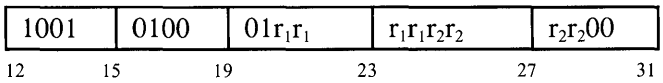
St $I \rightarrow R_1$ (32 bits - high half of 64)
(RP_1) converted to integer and rounded to 32 bits
(RP_1) preserved
Result $\rightarrow R_2$

Subtract Short (SUBS)



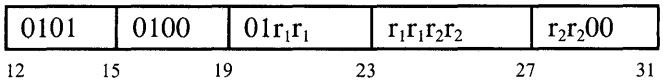
St $(R_2) - (R_1) \rightarrow R_2$

Subtract Immediate Short (SUBIS)



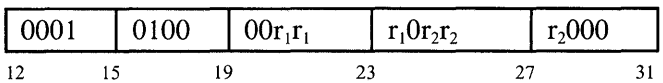
St $I \rightarrow R_1$
$(R_2) - (R_1) \rightarrow R_2$

Subtract Short Immediate (SUBSI)



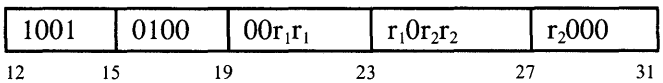
St $I \rightarrow R_2$
$(R_2) - (R_1) \rightarrow R_2$

Subtract Long (SUBL)



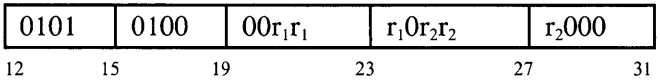
St $(RP_2) - (RP_1) \rightarrow RP_2$

Subtract Immediate Long (SUBIL)



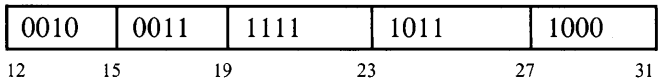
St $I \rightarrow R_1$ (32 bits — high half of 64)
$(RP_2) - (RP_1) \rightarrow RP_2$

Subtract Long Immediate (SUBLI)



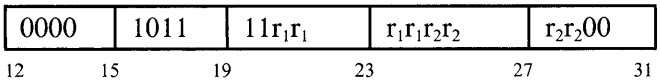
St $I \rightarrow R_2$ (32 bits — high half of 64)
$(RP_2) - (RP_1) \rightarrow RP_2$

Task Switch Unlock (TSKSWU) - Privileged Instruction



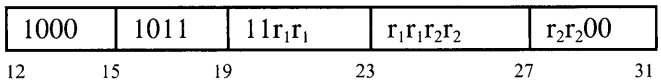
St, Su
- I Bits 27-31 → TASK ID REG (See Figure 1)
- Task Status (register X'E') → hardware status logic
- FPA board "unlocks" allowing operation on new task with previous status restored.

Truncate Short To Word (TSW)



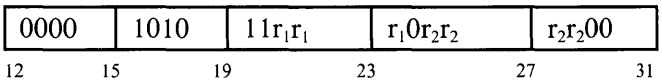
(R_1) converted to integer and truncated to 32 bits
(R_1) preserved
Result → R_2

Truncate Immediate Short To Word (TISW)



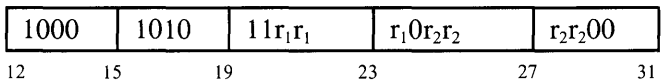
$I \rightarrow R_1$
(R_1) converted to integer and truncated to 32 bits
(R_1) preserved
Result → R_2

Truncate Long To Word (TLW)



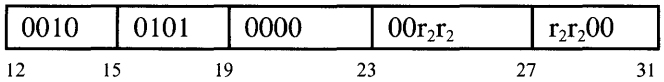
(RP_1) converted to integer and truncated to 32 bits
(RP_1) preserved
Result → R_2

Truncate Immediate Long To Word (TILW)



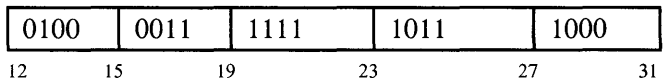
St $I \rightarrow R_1$ (32 bits - high half of 64)
(RP_1) converted to integer and truncated to 32 bits
(RP_1) preserved
Result → R_2

Write Float Register (WTFR)



St $I \rightarrow R_2$ (valid R_2 = X' 0-D')
(for long operands write high 32 bits at R_2; perform WTFR again with low 32 bits at $R_2 + 1$)

Write Status Register (WTSTR)



St $I \rightarrow$ Active Status Register (X'E')
$I \rightarrow$ Hardware Status Logic.

(Some bits unwritable — see status register description)

FPA Coding Example

Assume that general registers 8 and 9 contain two single precision floating-point numbers which are to be added together. The result is left in general register 2 in the same format. General register 15 is used for building the floating-point commands. Floating-point registers 2 and 3 will hold the operands.

```
cau     15,0xFF02       /* upper half of write float register                  /*
st      8,0x5008(15)    /* general register 8 to float register 2              /*
cau     15,0xFF05       /* upper half of Add Short Immediate                   /*
st      9,0x048C(15)    /* general register 9 to float register 3, 2 added to 3,  /*
                        /* result in 3                                         /*
cau     15,0xFF03       /* Read float register command field. Will borrow so   /*
                        /* use FF03 instead of FF02                            /*
ld      2,0xF0C0(15)    /* Float register 3 to general register 2              /*
```

FPA Programming Considerations

- Status Related
  - Exception type 5 (operand error) results if (1) a numeric operation is performed using reserved operands (infinity, not-a-number, or denormalized numbers), (2) result would yield reserved operand or (3) both operands of divide are zero. The results register is not changed in these cases.
  - Illegal command exception (type 7) occurs on all illegally coded FPA opcodes or when normal instructions are received when locked.
  - Floating point task switching is indicated at the board level by some combination of TSKSWU and LOCK instructions. However, status is not automatically saved when these instructions occur. Therefore, an RDSTR instruction must always precede the task switch to update the register file status register.
  - The RDFR instruction, with OP1 = E, should not be used to access FPSR (status) as this will not yield meaningful results; only RDSTR or RDSCX should be used.
  - The RDSCX instruction should be used only by the program check handler and during initialization. Other uses may result in loss of capability to recover from program checks.
  - The RDER instruction normally is used only by the program check handler. Other uses of this instruction may result in unpredictable results.
  - Each register set (Task) must be initialized with the sequence TSKSWU, RDSCX, WTSTR. Power on state is locked.
  - The program check handler should issue RDSCX following program check; repeat until successful or time out.
  - Numeric exceptions (types 1-6) cause program check on the command following the one with the numeric exception. The second instruction is not executed.
  - The FPA cannot produce the IEEE unordered result on comparisons of not-a-numbers.
  - Once a program check occurs, any other instructions except RDSCX will cause program check with the status register unchanged unless a bus error occurs.
- Real Mode Related
  - The system is normally used in the virtual mode of operation. If real mode operation is required, the following software restrictions apply:
    1. Any FPA or IOCC instruction must be followed by an instruction which uses the result of that load before a subsequent FPA instruction is executed.
    2. Any FPA store instruction must be followed by at least one store or two loads which reference system memory before a subsequent FPA instruction is executed.

3. Any FPA or IOCC instruction which follows an FPA load must be preceded by an instruction which uses the result of that load.

- Reset Related
  - Wait 6.5 μsec after power on reset or any channel reset before issuing more commands.

Advanced Floating-Point Accelerator

The RT PC Advanced Floating-Point Accelerator (AFPA) is an optional system unit feature which can be installed in the IBM 6151 or IBM 6150. It requires Version 2.1 Operating System software release or higher and will run programs written for the Floating-Point Accelerator (FPA) without recompiling or relinking, provided the restrictions listed in "AFPA Programming Considerations" on page 4-88 have been met. The AFPA provides substantially improved performance over the FPA. The AFPA also supports new enhanced functions and opcodes which require re-linking or recompilation.

The AFPA utilizes a dedicated slot on the system board. Operands may be stored on the AFPA and operations may be initiated by issuing processor load and store instructions. If the RT PC has an Advanced Processor Card, data may also be transferred using enhanced Direct Memory Access (DMA), further enhancing performance.

The AFPA operates independently of the system unit so that the system unit and AFPA operations can overlap. The board implements a subset of the IEEE 754 Standard for Binary Floating-Point Arithmetic (ANSI/IEEE STD 754-1985). Software is required to fully implement the standard. The board uses the Analog Devices ADSP-3210 Floating-Point Multiplier and ADSP-3221 Floating-Point ALU.

Block Diagram

Figure 4-7 on page 4-43 is a block diagram of the Advanced Floating-Point Accelerator board. When the system unit executes either a load or store instruction having any address in segment FF, the processor board routes the address and any associated data to the AFPA over the 32 bit bus. The AFPA decodes the command from the address.

Store instructions with addresses in segment FE are DMA instructions. Decoding is done in part by the AFPA and in part by the enhanced DMA interface on the Advanced Processor Card. Load instructions in segment FE are illegal, and should not be used for any reason.

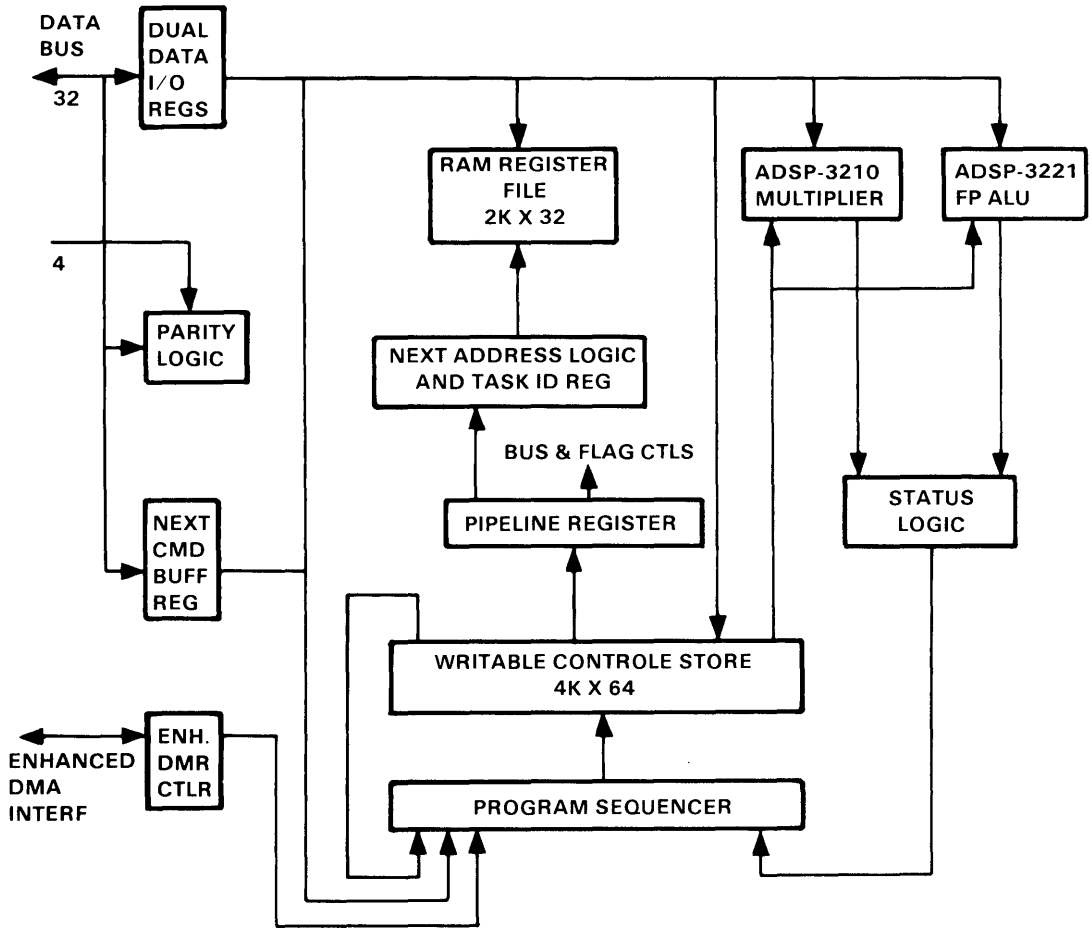


Figure 4-7. Advanced Floating-Point Accelerator Block Diagram

Data Formats

The AFPA supports two floating-point data formats; the 32-bit single (short) and the 64-bit double (long), as shown in Figure 4-8 amd Figure 4-9. The Advanced Floating-Point Accelerator performs arithmetic in both formats, converts between formats, and converts to and from the twos complement binary format. The S, EXP, and f are binary values where the leftmost bit is the most significant.

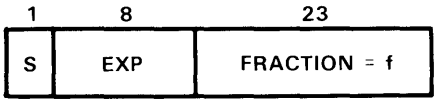


Figure 4-8. Single (Short) Precision Format

In single (short) precision format, the floating point number = $(-1)^s * (1.f) * 2^{EXP-127}$.

Figure 4-9. Double (Long) Precision Format

In double (long) precision format, the floating point number = $(-1)^s * (1.f) * 2^{EXP-1023}$.

Register Sets

The register file is a 2048 x 32 RAM which is organized as 32 sets of 64 registers, with each register containing 32 bits. Some of these sets are reserved for use by the microcode and are not available to the operating system. Initially, eight sets (register groups 24-31) are reserved, however the operating system should be able to accommodate a different number. The reserved sets are used for the following purposes:

- Temporary storage
- Constants

- Floating-point status register (FPASR) - One per register set
- Floating-point instruction register (FPIR) - One per register set.

Long operands are stored with the high order 32 bits at an even register address and with the low 32 bits at the next higher odd register address. Context switching is achieved by issuing the Task Switch Unlock command which loads the Task ID register (TIR). The TIR provides the most significant five bits of addressing.

All RAM locations within a register set are accessible via the register read and write commands. It is the responsibility of the operating system to prohibit access to the reserved sets.

Register Usage

The AFPA employs a two operand addressing scheme. Operand 1 (OP1) is the source operand address and Operand 2 (OP2) the destination operand address. OP1 and OP2 are normally different for instructions having two operands. They may be the same or different for instructions having only one operand. If the instruction produces a result it always overlays the data in OP2. The original contents of OP2 can be restored to facilitate exception handling.

Control Store

The control store is a 4096 x 64-bit RAM into which horizontal microcode is loaded to run a program sequencer and control other logic functions. Special commands are supported for reading and writing this RAM. This manual assumes that the control store has been loaded with the microcode algorithms which support the instructions described.

Instruction Description

All transfers of commands and data between the system processor and the AFPA are initiated by the system processor and accomplished by Store and Load instructions using memory mapped I/O. Two modes of data transfer are provided; programmed I/O (PIO) and Direct Memory Access (DMA).

PIO Mode

In the PIO mode memory addresses starting with X'FF' are decoded by the AFPA as commands to it, and are ignored by other devices on the processor channel. The remaining 24 bits of the 32-bit memory address are used as the AFPA command. If there is data associated with the command, it is transferred in the 32-bit data request packet associated with the Store instruction or the 32-bit reply packet associated with the Load instruction. If the command requires no data, the data associated with the request packet is discarded. The system processor address and AFPA command format is shown in Figure 4-10.

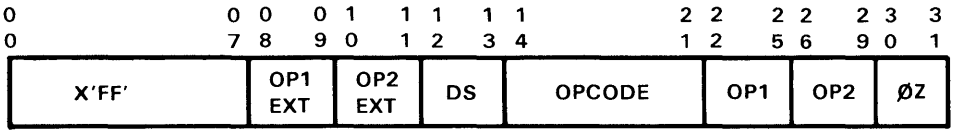


Figure 4-10. AFPA PIO Command Format

Bits 0 through 7 are the AFPA address for PIO and are fixed at X'FF'. Bits 8, 9 and 22 through 25 are the OP1 address. Bits 10, 11 and 26 through 29 are the OP2 address. Bits 12 and 13 indicate if the immediate data packet is to be used and whether it is to go into the register indicated by OP1 or OP2. Bits 14 through 21 are the op-code field. Bit 31 is always 0 except for some compare instructions.

Data Source (DS) Definitions

Bits 12-13	Function
00	No immediate data
01	Immediate data to OP2 address
10	Immediate data to OP1 address
11	Reserved

Bits 0 through 3 do not actually reach the AFPA and bits 4 through 7 are sent to AFPA as address extension bits. Bits 0 through 7 however, are replaced in the 32-bit instruction which reaches the AFPA with the 8-bit processor channel control word, providing AFPA control information. The two operand fields, OP1 and OP2, specify the AFPA source and destination addresses, respectively.

Most floating-point instructions are unprivileged instructions. Task Switch/Unlock, Lock, Disable Control Store, Read/Write Control Store and Read Status/Clear Exception are valid only in privileged state.

AFPA must be used in the virtual mode of operation in which program checks (exception conditions) are reported on both Load and Store instructions. AFPA exceptions are caused by parity errors, illegal or nonsupported commands, or various IEEE boundary situations, such as overflow. Operation in the real mode is not supported.

Status Register (FPASR)

The AFPA status register, shown in Figure 4-11 is a hardware register used to control the AFPA operation, (such as rounding mode) and to identify various exception conditions. The last status register state is saved in a register location corresponding to the active register set in one of the reserved register sets.

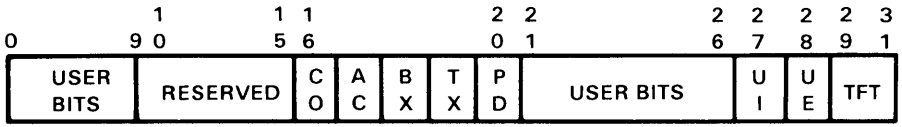


Figure 4-11. Floating Point Status Registers

The AFPA status register consists of two logical parts:

- User bits (Bits 0-9, 21-26) - Part of user application interface.
  - Mode control bits (Bits 0, 3, 5, 7, 9, 23-24, 26) — These bits are set and reset by the user, or in some cases by VRM.
  - Status bits (Bits 1, 2, 4, 6, 8, 21-22, 25) — These bits are set by AFPA or the Virtual Resource Manager (VRM), but they are cleared by the user. The user accesses these bits to determine if IEEE exceptions occurred.
- Hardware bits (Bits 16-20, 27-31) - Transparent to user at application interface.
  - Mode control bits (Bits 20, 28) — These bits are under control of the operating system. Bit 20 is normally set to 0 and bit 28 is normally set to 1.
  - Status bits (Bits 16-19, 27, 29-31) — These bits are normally set and cleared by the AFPA. However, in certain cases, they can be set and cleared by the operating system.

Hardware Status Bit Definitions

Control Bits	Function
20	Program Check Disable 1 = Diagnostic purposes 0 = Normal setting
28	Underflow Enable 1 = Normal setting 0 = Unpredictable results can occur.
Status Bits	**Function**
10-15	Reserved
16	Conforming Operation (Always 0 for FPA) 1 = Conforming operation, no operating system intervention required 0 = Nonconforming operation, operating system intervention required in order to yield IEEE compatible results or flags.
17	1 = Aborted Command Board was reset while busy; set by hardware; reset only by RDSCX command. Does not cause program check. See command list for RDSCX.
18	Bus Exception 1 = Program check due to bus parity error on last AFPA command. Set by hardware; reset by RDSCX. Nonmaskable.
19	Task Exception 1 = Task Exception; set by hardware or operating system, indicates that an AFPA exception is causing a program check, reset by RDSCX. Set on last command if a locked exception is detected, set on second to last command for all other exceptions.
27	Hardware Underflow Indication 1 = Underflow occurred since this bit was written to zero. 0 = No underflow since this bit was written to zero.

29-31 Task Fault Type (TFT)

These bits are usually changed on each command performed regardless of bits 20, 26 or 28.

Type	29	30	31	
0	0	0	0	No Exception
1	0	0	1	Underflow
2	0	1	0	Overflow
3	0	1	1	Divide by zero
4	1	0	0	DMA write data parity error
5	1	0	1	Invalid Operation
6	1	1	0	Inexact Result
7	1	1	1	Illegal Command — reset only with RDSCX; Nonmaskable

User Status Bit Definitions

Control Bits	Function
0	Kill 0 = All user traps disabled 1 = User traps enabled
3	I/O (Invalid operation) Trap enable
5	DZ (Divide by zero) Trap enable
7	OF (Overflow) Trap enable
9	UF (Underflow) Trap enable
26	IR (Inexact result) Trap enable For bits 3, 5, 7, 9, 26: 0 = Specific user trap type is disabled. 1 = Specific user trap type is enabled and corresponding trap handler is provided by the user.
23-24	Rounding Mode (RM) 00 = Round (nearest) 01 = Truncate (towards zero) 10 = Floor (towards - ∞) 11 = Ceiling (towards + ∞)

Status Bits	Function
1	XCP-FLAG Set to 1 when any Flag is set to 1 Reset by user or OS.
2	I/O (Invalid operation) Flag
4	DZ (Divide by zero) Flag
6	OF (Overflow) Flag
8	UF (Underflow) Flag
25	IR (Inexact result) Flag For bits 2, 4, 6, 8, 25: 0 = Specific condition did not occur 1 = Specific condition did occur.
21,22	Compare result (N,Z) 00 = Op at source address < Op at destination address 01 = Op at source address = Op at destination address 10 = Op at source address > Op at destination address 11 = Unordered compare due to either operand being a NaN (Not a Number).

AFPA Instruction Register (FPIR)

When an exception is reported, a register location in one of the reserved register sets corresponding to the active register set will contain the following information about the last executed command. The first 8 bits are taken from the channel control word and the remaining bits are taken from the command.

Bit	Meaning
0	0 = Memory Mapped
1	0 = Load 1 = Store
2,3	10 = 4 Byte Operation
4	0 = Memory Protect Disabled
5	1 = Virtual Mode 0 = Should not be used
6	0 = Privileged 1 = Unprivileged (Normal) State
7	0 = DMA instructions 1 = PIO instructions
8-31	Bits 8-31 from the command.

Command Encoding (Effective Address of Load or Store)

r_1, r_2	Binary weighted bits specifying one of the 64 active short registers or 32 active long registers pairs as operand 1 or 2.
R_1,R_2	Address of 32-bit register, specified by r_1,r_2.
(R_1), (R_2)	Contents of register address R_1,R_2.
I	32 bits of immediate data associated with a store type command.
RP_1,RP_2	Address of 64-bit register pair specified by r_1,r_2. Least significant bit of r_1,r_2 must be 0.
(RP_1), (RP_2)	Contents of register pair RP_1,RP_2.
Su	Indicates that exception will result if system is not in supervisory state.
Ld	Load type instruction.
St	Store instruction.
Boxed area	Bits 8-31 of the address of the load or store instruction. Bits 0-7 are always X'FF' (PIO) or X'FE' (DMA).
→	Replaces the content of.

Basic PIO Floating-Point Instructions

The floating-point instructions corresponding to, or required for, those provided with the original FPA are described and listed in alphabetical order. Commands that have an immediate form are listed with the normal form. The Z variable in the TS field is set to 0 in PIO mode.

Notice that these instructions have OP1 EXT (OP1 address extension) and OP2 EXT (OP2 address extension) fields. For those instructions compatible with the original FPA these bits are always 'all zeros' thereby supporting only 16 registers per user. For AFPA, these bits are usable, thereby supporting the full 64 registers per user.

Absolute Short (ABSS, ABSIS)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01110101	$r_1r_1r_1r_1$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ if X⅛
$| (R_1) | \rightarrow R_2$
(R_1) preserved

Absolute Long (ABSL, ABSIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01110100	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
$| (RP_1) | \rightarrow RP_2$
(RP_1) preserved

Add Short (ADDS, ADDIS, ADDSI)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	xy	01000001	$r_1r_1r_1r_1$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ if XY = 10
$I \rightarrow R_2$ if XY = 01
$(R_1) + (R_2) \rightarrow R_2$
(R_1) preserved

Add Long (ADDL, ADDIL, ADDLI)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	xy	01000000	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if XY = 10
$I \rightarrow R_2$ (32 bits - high half of 64) if XY = 01
$(RP_1) + (RP_2) \rightarrow RP_2$
(RP_1) preserved

Compare Short (COMS, COMIS)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01001001	$r_1r_1r_1r_1$	$r_2r_2r_2r_2$	zk
08	10	12	14	22	26	30

St $I \rightarrow R_1$ if $X = 1$
If $K = 0$, non trapping compares
If $K = 1$, trapping compares
(R_1) compared to (R_2)
Status register N, Z set (see status register bits 21-22)
(R_1), (R_2) preserved

Compare Long (COML, COMIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01001000	$r_1r_1r_10$	$r_2r_2r_20$	zk
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if $X = 1$
If $K = 0$, non trapping compares
If $K = 1$, trapping compares
(RP_1) compared to (RP_2)
Status register N, Z set (see status register bits 21-22)
(RP_1), (RP_2) preserved

Convert Short To Long (CSL, CISL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	00011011	$r_1r_1r_1r_1$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits) if X = 1
(R_1) converted to long format (64 bits)
(R_1) preserved
Result $\rightarrow RP_2$

Convert Long To Short (CLS, CILS)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	00010110	$r_1r_1r_10$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
(RP_1) converted to short format (32 bits)
(RP_1) preserved
Result $\rightarrow R_2$

Convert Word To Short (CWS, CIWS)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	00000111	$r_1r_1r_1r_1$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32-bit integer) if X = 1
(R_1), integer word, converted to short float format
(R_1) preserved
Result $\rightarrow R_2$

Convert Word To Long (CWL, CIWL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	00000011	$r_1r_1r_1r_1$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32-bit integer) if X = 1
(R_1), integer word, converted to long float format
(R_1) preserved
Result $\rightarrow RP_2$

Copy Short (COPS)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	00	01000101	$r_1r_1r_1r_1$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $(R_1) \rightarrow R_2$
(R_1) preserved

Copy Long (COPL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	00	01000100	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $(RP_1) \rightarrow RP_2$
(RP_1) preserved

Divide Short (DIVS, DIVIS, DIVSI)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	xy	01100001	$r_1r_1r_1r_1$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ if XY = 10
$I \rightarrow R_2$ if XY = 01
$(R_2) \div (R_1) \rightarrow R_2$
(R_1) preserved

Divide Long (DIVL, DIVIL, DIVLI)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	xy	01100000	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if XY = 10
$I \rightarrow R_2$ (32 bits - high half of 64) if XY = 01
$(RP_2) \div (RP_1) \rightarrow RP_2$
(RP_1) preserved

Floor Short To Word (FSW, FISW)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	00111111	$r_1r_1r_1r_1$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ if $X = 1$
(R_1) converted to integer and floored to 32 bits
(R_1) preserved
Result $\rightarrow R_2$

Floor Long To Word (FLW, FILW)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	00111011	$r_1r_1r_10$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if $X = 1$
(RP_1) converted to integer and floored to 32 bits
(RP_1) preserved
Result $\rightarrow R_2$

Lock Floating Point (LOCKFP) - Privileged Instruction

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
00	00	00	10011100	0000	0000	00
08	10	12	14	22	26	30

St, Su Locks out all instructions except TSKSWU, RDSCX, LOCKFP, RCS, WCS, DISCS (TFT = 111 exception generated upon receiving any instruction other than those listed).

Multiply Short (MULS, MULIS, MULSI)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	xy	01110001	$r_1r_1r_1r_1$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ if XY = 10
$I \rightarrow R_2$ if XY = 01
$(R_1) * (R_2) \rightarrow R_2$
(R_1) preserved

Multiply Long (MULL, MULIL, MULLI)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	xy	01110000	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if XY = 10
$I \rightarrow R_2$ (32 bits - high half of 64) if XY = 01
$(RP_2) * (RP_1) \rightarrow RP_2$
(RP_1) preserved

Negate Short (NEGS, NEGIS)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01010101	$r_1r_1r_1r_1$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ if X = 1
$(R_1) * (-1) \rightarrow R_2$
(R_1) preserved

Negate Long (NEGL, NEGIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01010100	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
$(RP_1) * (-1) \rightarrow RP_2$
(RP_1) preserved

Read Float Register (RDFR)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	00	00	10111100	$r_1r_1r_1r_1$	0000	zz
08	10	12	14	22	26	30

Ld $(R_1) \rightarrow$ CPU (for 64 bits perform second RDFR at $R_1 + 1$).

Read Exception Register (RDER) - Not Supported

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
00	00	00	10111100	1111	0000	00
08	10	12	14	22	26	30

Ld This instruction is not supported by AFPA. See description of "Read Instruction Register (RDIR)" on page 4-71.

Read Status Register (RDSTR)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
00	00	0x	00110111	1110	1110	00
08	10	12	14	22	26	30

Ld Hardware status → Status Reg; Then Status Reg → CPU
Notice that DS can equal 00 or 01 depending on the compiler version used. The same function is performed in either case.

Read Status Clear Exception (RDSCX) - Privileged Instruction

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
00	00	00	10110111	1110	1110	00
08	10	12	14	22	26	30

Ld, Su Hardware status → Status Reg of active register set; Then Status Reg → CPU

- Program check conditions and associated hardware status bits reset
- Normally used only at initialization or following an exception.

Round Short To Word (RSW, RISW)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	00100111	$r_1r_1r_1r_1$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ if $X = 1$
(R_1) converted to integer and rounded to 32 bits
(R_1) preserved
Result → R_2

Round Long To Word (RLW, RILW)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	00100011	$r_1r_1r_10$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if $X = 1$
(RP_1) converted to integer and rounded to 32 bits
(RP_1) preserved
Result $\rightarrow R_2$

Subtract Short (SUBS, SUBIS, SUBSI)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	xy	01010001	$r_1r_1r_1r_1$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ if $XY = 10$
$I \rightarrow R_2$ if $XY = 01$
$(R_2) - (R_1) \rightarrow RP_2$
(R_1) preserved

Subtract (SUBL, SUBIL, SUBLI)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	xy	01010000	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if XY = 10
$I \rightarrow R_2$ (32 bits - high half of 64) if XY = 01
$(RP_2) - (RP_1) \rightarrow RP_2$
(RP_1) preserved

Task Switch Unlock (TSKSWU) - Privileged Instruction

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
00	00	00	10001111	1110	1110	00
08	10	12	14	22	26	30

St, Su
- I Bits 27-31 → Task ID Reg (See Figure 1)
- Previous Task Status → Status Register
- AFPA board 'unlocks' allowing operation on new task with present status restored.
- If an exception is pending or is generated by an operation in process, it is stored in previous task's status register and taken when the previous task is reactivated. The TSKSWD will not have to be reissued.

Truncate Short To Word (TSW, TISW)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	00101111	$r_1r_1r_1r_1$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ if $X = 1$
(R_1) converted to integer and truncated to 32 bits
(R_1) preserved
Result $\rightarrow R_2$

Truncate Long To Word (TLW, TILW)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	00101011	$r_1r_1r_10$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if $X = 1$
(RP_1) converted to integer and truncated to 32 bits
(RP_1) preserved
Result $\rightarrow R_2$

Write Float Register (WTFR)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
00	r_2r_2	00	10010100	0000	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_2$ (valid R_2 = X '0-3F') For long operands write high 32 bits at R_2; perform WTFR again with low 32 bits at $R_2 + 1$.

Write Status Register (WTSTR)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
00	00	01	00001111	1110	1110	00
08	10	12	14	22	26	30

St $I \rightarrow$ Active Task Status Register
$I \rightarrow$ Hardware Status Logic

(Some bits unwritable by user - see status register description)

Special Purpose PIO Floating Point Instructions

These instructions are used for exception handling, diagnostic purposes and for controlling access to control store. Although not part of the base function, they may be required as support for the base function.

Disable Control Store Access (DISCS) - Privileged Instruction

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
00	00	11	10011111	0000	0000	zz
08	10	12	14	22	26	30

Ld, Su Terminates control store access mode after IPL load of the control store.
No useful data contained in reply

Read Control Store (RCS) - Privileged Instruction

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
00	01	11	CS Address			0y
08	10	12	14	22	26	30

Ld, Su

- Control store specified by bits 14-29 → CPU
- If Y = 0, Read 32 most significant bits
- If Y = 1, Read 32 least significant bits
- Must be in control store access mode.

Read Instruction Register (RDIR)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
00	00	00	10111111	0000	0000	00
08	10	12	14	22	26	30

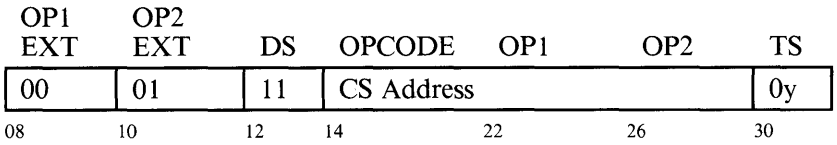
Ld Instruction Register → CPU
Contains failing instruction
Replaces RDER instruction.

Task Switch Unlock With Exception (TSKSWD) - Privileged Instruction

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
00	00	00	10011101	1110	1110	00
08	10	12	14	22	26	30

St, Su
- I Bits 27-31 → Task ID Reg (See Figure 1)
- Previous Task Status → Status register
- AFPA board 'unlocks' allowing operation on new task with present task status restored.
- If an exception is pending or is generated by an operation in process, a reply with exception is returned. The TSKSWD will be reissued after servicing the exception.

Write Control Store (WCS) - Privileged Instruction



St, Su
- I → Control store specified by bits 14-29
- If Y = 0, Write 32 most significant bits
- If Y = 1, Write 32 least significant bits
- Must be in control store access mode

Enhanced PIO Floating-Point Instructions

The following instructions are new functions that are provided by AFPA. These instructions, and all instructions listed in "Basic PIO Floating-Point Instructions" on page 4-52 which specify R_1, RP_1, R_2, or RP_2 as a variable, can use the entire addressing range provided by AFPA. This range consists of 32 double precision or 64 single precision addresses.

Arc Tangent Long (ATANL, ATANIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	11010100	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
$Atan(RP_1) \rightarrow RP_2$
(RP_1) preserved

Arc Tangent2 Long (ATAN2L, ATAN2IL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01010010	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
$Atan2(RP_1) \rightarrow RP_2$
(RP_1) preserved

Compare Trapping Short (COMTS, COMTIS)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01001011	$r_1r_1r_1r_1$	$r_2r_2r_2r_2$	00
08	10	12	14	22	26	30

St This opcode indicates exception for unordered status
$I \rightarrow R_1$, if $X = 1$
(R_1) compared to (R_2)
Status register N, Z set
(R_1), (R_2) preserved

Compare Trapping Long (COMTL, COMTIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01001010	$r_1r_1r_10$	$r_2r_2r_20$	00
08	10	12	14	22	26	30

St This opcode indicates exception for unordered status
$I \rightarrow R_1$, (32 bits - high half of 64) if $X = 1$
(RP_1) compared to (RP_2)
Status register N, Z set
(RP_1), (RP_2) preserved

Cosine Long (COSL, COSIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	11000010	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
$Cos(RP_1) \rightarrow RP_2$
(RP_1) preserved

*E**X Long (EXPL, EXPIL)*

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	11011000	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
$E^{**}(RP_1) \rightarrow RP_2$
(RP_1) preserved

Integer Part Long (INTL, INTIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	11001010	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
$Int(RP_1) \rightarrow RP_2$ where Int(X) = Integer part of 'X' in floating point format.
(RP_1) preserved

Log Base Long (LOGBL, LOGBIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01010110	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
Exponent of $(RP_1) \rightarrow RP_2$
(RP_1) preserved

Log Base 10 Long (LOG10L, LOG10IL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	11011110	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
$Log(RP_1) \rightarrow RP_2$, Log (X) = log base 10
(RP_1) preserved

Log Natural Long (LOGL, LOGIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	11011100	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
$LN(RP_1) \rightarrow RP_2$, LN (X) = natural log (X)
(RP_1) preserved

Partial Modulo Long (MODL, MODIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01000010	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
(RP_2) modulo $(RP_1) \rightarrow RP_2$
Result = (RP_2) - N * (RP_1), N is chosen to provide result < (RP_1)
(RP_1) preserved

Note: The Partial Modulo Long instruction may yield a final answer or a partial modulo result. In case of a partial modulo result, the instruction must be reissued by software until the final answer is obtained. If the result is partial, an asynchronous exception is generated. In the FPASR, CO = 0, TX = 1, and TFT = 110.

No-Op (NOOP)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	00	x0	10011111	$r_1r_1r_1r_1$	0000	00
08	10	12	14	22	26	30

St $I \rightarrow R$, If X = 1
Performs no FP function

Partial Remainder Long (REML, REMIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01100010	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
(RP_2) Remainder $(RP_1) \rightarrow RP_2$
Remainder = IEEE remainder, which is same as modulo except that result is least magnitude remainder
(RP_1) preserved

Note: The Partial Remainder Long instruction may yield a final remainder or a partial remainder. In case of a partial remainder, the instruction must be reissued by software until the final remainder is obtained. If the result is partial, a asynchronous exception is generated. In the FPASR, CO = 0, TX = 1, and TFT = 110.

Scale Binary Long (SCALBL, SCALBIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01110010	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
$(RP_2) * 2 \text{ exp } (RP_1) \rightarrow RP_2$
(RP_1) preserved

Sine Long (SINL, SINIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	11000000	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
$Sin(RP_1) \rightarrow RP_2$
(RP_1) preserved

Square Root Short (SQRS, SQRIS)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01100101	$r_1r_1r_1r_1$	$r_2r_2r_2r_2$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ if X = 1
Square root $(R_1) \rightarrow R_2$
(R_1) preserved

Square Root Long (SQRL, SQRIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	01100100	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
Square root $(RP_1) \rightarrow RP_2$
(RP_1) preserved

Tangent Long (TANL, TANIL)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	r_2r_2	x0	11000100	$r_1r_1r_10$	$r_2r_2r_20$	zz
08	10	12	14	22	26	30

St $I \rightarrow R_1$ (32 bits - high half of 64) if X = 1
Tan $(RP_1) \rightarrow RP_2$
(RP_1) preserved

Write Floating Point Instruction Register (WTIR)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
00	00	00	10111011	0000	0000	00
08	10	12	14	22	26	30

St I → Active Task Instruction Register

Enhanced DMA Instructions X'FE'

All basic and enhanced floating point instructions that meet the following criteria can transfer operands via Enhanced Direct Memory Access (DMA), which improves performance;

- Must be unprivileged instruction
- Instruction must previously be defined as one of the following:
  - A Store containing user operands as immediate data
  - A Load whose result contains user data.

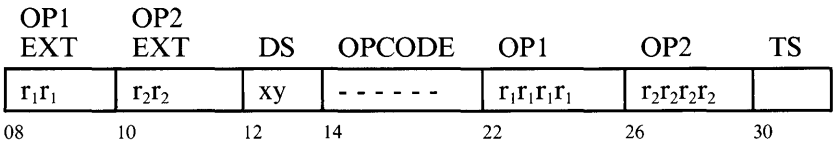
In this section, sequential implies that execution of the DMA related instruction will not occur until after the completion of the previous instruction.

The 'WW' in the DMA instruction's TS field is defined as follows:

TS	Function
00	Reserved
01	Single precision - one word of DMA data
10	Double precision - two words of DMA data
11	Multiple - 'N' words of DMA data

Note: The value 'N' is set under program control by executing a special X'FE' instruction. See "DMA Write Length Register (WRNL)" on page 4-86.

DMA Write Register Sequential Instructions



Opcode — Same as used in any unprivileged PIO instruction listed in "Basic PIO Floating-Point Instructions" on page 4-52 and "Enhanced PIO Floating-Point Instructions" on page 4-73, except for the following:

COPS, COPL - No immediate data involved
WTFR - Use opcode = 9F for this case if DMA used
WTSTR - Must use PIO instruction.

I = Effective address of initial memory location
XY = 10 Sequential transfer from system memory to FP registers starting at address specified by R_1
XY = 01 Sequential transfer from system memory to FR registers starting at address specified by R_2

Note: Write synchronization restriction; if this DMA instruction is followed by a processor store to the same system memory location as involved in the DMA WR instruction, it will be necessary to insert a synchronizing instruction between the two operations in order to avoid incorrect results. The synchronizing instruction can be a NO-OP.

DMA Read Register Sequential (RDDMA)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
r_1r_1	11	11	10011111	$r_1r_1r_1r_1$	0000	ww
08	10	12	14	22	26	30

I = Effective address of initial memory location.
Sequential transfer from FP registers to system memory, starting at FP register address specified by R_1
This opcode (NO OP) is the only DMA read supported.

Note: Read synchronization restriction; if the data being transferred into system memory by this instruction must be transferred into the processor's general purpose registers, it is necessary to insert an FPA synchronization instructions (such as NO OP) between the DMA read and the following loads to avoid incorrect data.

DMA Write Length Register (WRLN)

OP1 EXT	OP2 EXT	DS	OPCODE	OP1	OP2	TS
00	01	11	10011111	0000	0000	11
08	10	12	14	22	26	30

Data packet

I (least significant 5 bits) → Length register on processor board.

I bits 0 - 25 = zero

I bits 26 - 31 = Specify DMA length of N,

If X'00', N = 64
If X'01', N = 1
If X'3F', N = 63

This instruction will not execute if AFPA has an exception pending from the previous floating point instruction.

AFPA Coding Example

Assume that general registers 8 and 9 contain two single precision floating-point numbers that are to be added together. The result is left in general register 2 in the same format. General register 15 is used for building the floating-point commands. Floating-point registers 2 and 3 will hold the operands.

```
cau     15,0xFF02       /* upper half of write float register                  /*
st      8,0x5008(15)    /* general register 8 to float register 2              /*
cau     15,0xFF05       /* upper half of Add Short Immediate                   /*
st      9,0x048C(15)    /* general register 9 to float register 3, 2 added to 3,  /*
                        /* result in 3                                         /*
cau     15,0xFF03       /* Read float register command field. Will borrow so   /*
                        /* use FF03 instead of FF02                            /*
ld      2,0xF0C0(15)    /* Float register 3 to general register 2              /*
```

AFPA Programming Considerations

- Compatibility Related

  Applications compiled with Release 1.1 compilers using the direct option should run correctly on AFPA without change assuming the following instructions have not been used.

  - RDER or RDFR (with $R_1 = F$) instructions as defined for FPA. For AFPA, these instructions have been replaced by RDIR for reading the Floating Point Instruction Register (FPIR).
  - RDFR instruction (with $R_1 = E$) where the intent is to read the contents of the status register (FPASR) on the FPA. Use of this instruction for this purpose is not supported on FPA. For AFPA, this instruction will input the contents of the E register, not the contents of FPASR. Rather, RDSTR and RDSCX instruction input the contents of FPASR for FPA and AFPA.
  - WTFR instruction (with $R_2 = F$) where the intent is to write user data into the Floating Point Instruction Register (FPIR) on FPA. Use of this instruction for this purpose is not supported for FPA. For AFPA, this instruction writes the user data into the F user register, not the FPIR. Rather, the WTFR instruction can be used to write FPIR for AFPA.
  - WTFR instruction (with $R_2 = E$) where the intent is to write user data into the status register (FPASR) on FPA. Use of this instruction for this purpose is is not supported for FPA. For AFPA, this instruction will write the user data into the E user register, not the FPASR. Rather, the WTSTR instruction is provided both for FPA and AFPA for writing FPASR.

  Existing assembly code should also run without change assuming the previously listed restrictions have been met as well as the following restriction:

  The program must not execute an RDSTR or RDSCX instruction and expect the CO bit (bit 16) to be 0. FPA will always return 0 for bit 16 of FPASR. However, AFPA can return a 0 or a 1 in this bit position.

- Status Related
  - Exception type (TFT = 101) may result if:
    - — A numeric is performed using reserved operands (not-a-number, or denormalized numbers)
    - — Result is not defined for operand/operation combination
    - — Both operands of divide are zero.

- Illegal command exception (TFT = 111) occurs on some illegally coded FPA instructions, on some bus errors or when normal instructions are received when locked.
- Floating point task switching is indicated at the board level by some combination of TSKSWU and LOCK instructions. However, status is automatically saved when these instructions occur.
- The RDSCX instruction should be used only by the program check handler and during initialization. Other uses may result in loss of capability to recover from program checks.
- The RDIR instruction normally is used only by the program check handler. Other uses of this instruction may result in unpredictable results.
- Each register set (Task) must be initialized with the sequence TSKSWU, RDSCX, WTSTR. Power on state is locked.
- The program check handler should issue RDSCX following program check; repeat until successful or time out.
- Numeric exceptions (TFT = 1-3,5-6) cause program check on the command following the one with the numeric exception. The second instruction is not executed.
- Once a program check occurs, any other instructions except RDSCX will cause program check with the status register unchanged unless a bus error occurs.

• Real Mode Related

  The system is normally used in the virtual mode of operation. Real mode operation is not recommended or supported.

• Reset Related

  - Wait 6.5 μsec after power on reset or any channel reset before issuing more commands.

Floating-Point Accelerator Board Pin Assignments

The following tables contain the Floating-Point Accelerator board pin assignments.

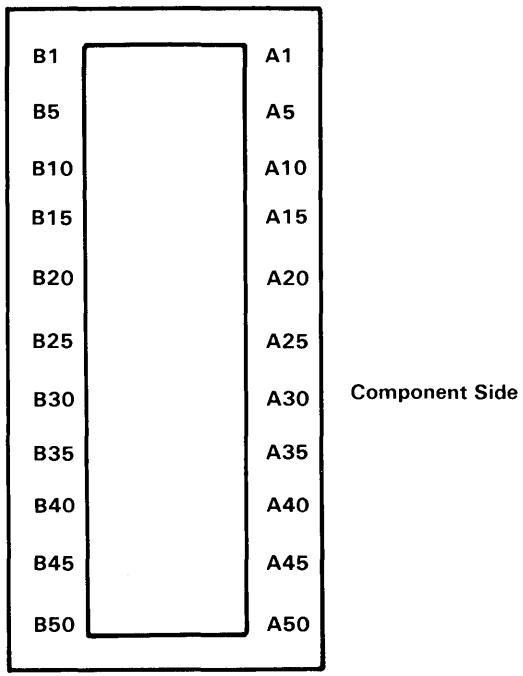


Figure 4-12. 100 Pin FPA Board Slot

I/O Pin	Signal Name I/O
A 1	+ 5 Vdc
A 2	+ 5 Vdc
A 3	+ ADR/DATA 00
A 4	+ ADR/DATA 01
A 5	+ ADR/DATA 02
A 6	+ ADR/DATA 03
A 7	+ ADR/DATA 04
A 8	+ ADR/DATA 05
A 9	+ ADR/DATA 06
A 10	+ ADR/DATA 07
A 11	+ ADR/DATA 08
A 12	+ ADR/DATA 09
A 13	GND
A 14	+ ADR/DATA 10
A 15	+ ADR/DATA 11
A 16	+ ADR/DATA 12
A 17	+ ADR/DATA 13
A 18	+ ADR/DATA 14
A 19	+ ADR/DATA 15
A 20	+ ADR/DATA 16
A 21	+ ADR/DATA 17
A 22	+ ADR/DATA 18
A 23	+ ADR/DATA 19
A 24	GND
A 25	GND

Figure 4-13 (Part 1 of 2). Floating-Point Accelerator Slot (A-Side)

I/O Pin	Signal Name I/O
A 26	+ ADR/DATA 20
A 27	+ ADR/DATA 21
A 28	+ ADR/DATA 22
A 29	+ ADR/DATA 23
A 30	+ ADR/DATA 24
A 31	+ ADR/DATA 25
A 32	+ ADR/DATA 26
A 33	+ ADR/DATA 27
A 34	+ ADR/DATA 28
A 35	+ ADR/DATA 29
A 36	+ ADR/DATA 30
A 37	GND
A 38	+ ADR/DATA 31
A 39	Reserved
A 40	-RSC HOLD
A 41	-IREQ 0
A 42	-IREQ 2
A 43	-IREQ 4
A 44	Reserved
A 45	Reserved
A 46	Reserved
A 47	GND
A 48	GND
A 49	+ 5 Vdc
A 50	+ 5 Vdc

Figure 4-13 (Part 2 of 2). Floating-Point Accelerator Slot (A-Side)

I/O Pin	Signal Name I/O
B 1	+ 5 Vdc
B 2	+ 5 Vdc
B 3	+ ADR EXT 4
B 4	+ ADD EXT 5
B 5	+ ADR EXT 6
B 6	+ ADD EXT 7
B 7	-FPA ID 5
B 8	-Channel End
B 9	GND
B 10	+ Parity 0
B 11	-Channel Reset
B 12	-POR
B 13	GND
B 14	-Busy
B 15	-Ready
B 16	-Channel Error
B 17	-I/O Cycle
B 18	-IOCC Active
B 19	+ Parity 1
B 20	GND
B 21	-Data Gate
B 22	-FPA ID 6
B 23	-Data Strobe
B 24	GND
B 25	GND

Figure 4-14 (Part 1 of 2). Floating-Point Accelerator Slot (B-Side)

I/O Pin	Signal Name I/O
B 26	-Timer In
B 27	-HLDA
B 28	-HRQ
B 29	+ Parity 2
B 30	GND
B 31	-FPA ID 7
B 32	-FPA Busy
B 33	+ Tag
B 34	GND
B 35	+ Exception
B 36	-Sys Reset
B 37	GND
B 38	+ Parity 3
B 39	-PIO Request
B 40	-Address Gate
B 41	-IREQ 1
B 42	-IREQ 3
B 43	Reserved
B 44	-I/O Trap
B 45	-DMA Req
B 46	-DMA Cycle
B 47	GND
B 48	GND
B 49	+ 5 Vdc
B 50	+ 5 Vdc

Figure 4-14 (Part 2 of 2). Floating-Point Accelerator Slot (B-Side)

Section 5. System Boards

CONTENTS

About this Section

This section contains information about the IBM 6151 and IBM 6150 system boards. The various addressing modes and translation modes are discussed. Direct Memory Access, real time clock and serial ports are described. The keyboard, locator and speaker adapter operations are described in detail.

IBM 6151 and IBM 6150 Common Features

Components

- Eight channel direct memory access (DMA)
- DMA address translation
- Fifteen-level interrupt
- Keyboard, locator, speaker adapter
- Real-time clock with CMOS RAM and battery backup
- Processor to I/O channel conversion
- System control registers.

Attachments

- Processor board slot
- Floating-Point Accelerator option slot
- Two system memory slots
- Keyboard and speaker
- Locator
- Operator panel.

IBM 6150 Unique Features

The IBM 6150 system board fits vertically in the base of the system unit and is approximately 29.2 by 43.2 centimeters (11.5 by 17 inches). It is a multilayer board with internal ground and +5 Vdc planes, plus 4 signal planes using a 2 land per channel design. It has the additional feature of 2 RS232C serial ports with DMA transmit capability.

There are 8 I/O slots on the IBM 6150 system board.

- Six with a 62-pin and a 40-pin board-edge slot
- Two with a 62-pin board-edge slot.

Slot	Unique	Connectors	Assignment
1	No	62-Pin/40-Pin	Disk/Diskette Adapter
2	No	62-Pin/40-Pin	Option
3	No	62-Pin	Option
4	No	62-Pin/40-Pin	Option
5	No	62-Pin/40-Pin	Option
6	No	62-Pin	Option
7	No	62-Pin/40-Pin	Option
8	Yes	62-Pin/40-Pin	Coprocessor/Option

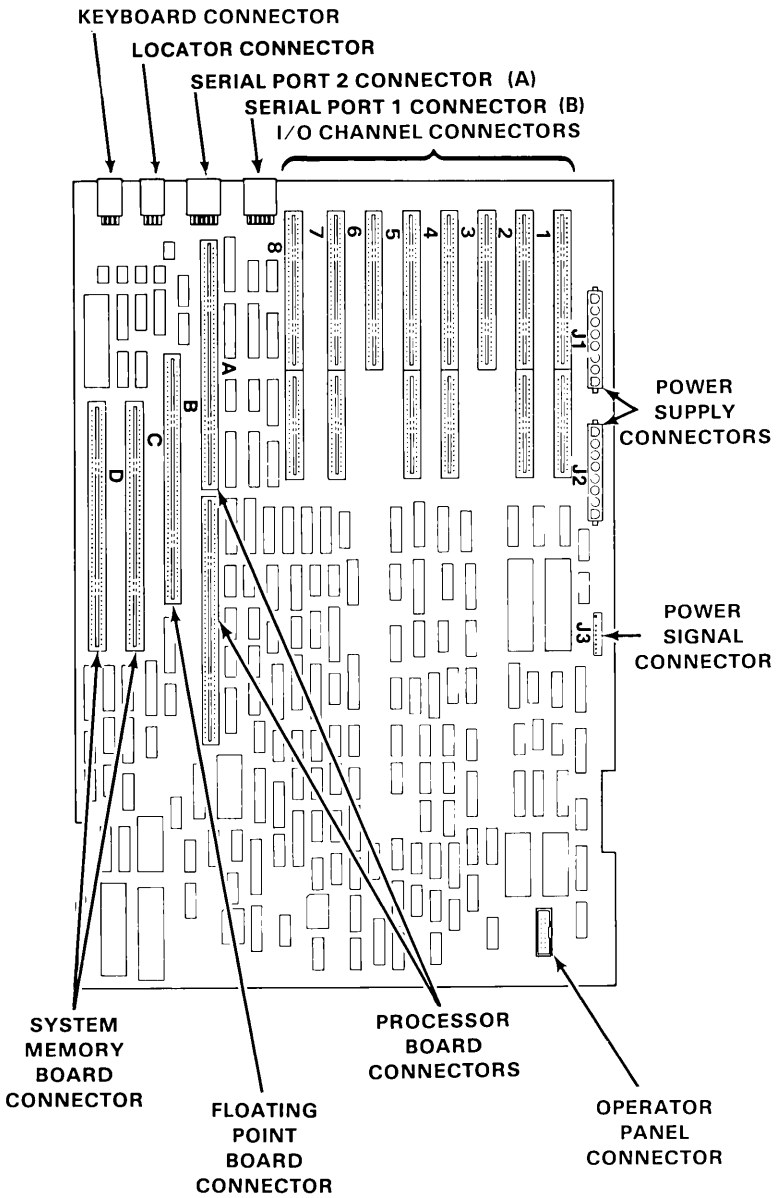


Figure 5-1. IBM 6150 Connector Locations

IBM 6151 Unique Features

The IBM 6151 system board fits horizontally in the base of the system unit and is approximately 30.5 by 35.5 centimeters (12 by 14 inches). It is a multilayer board with internal ground and +5 Vdc planes, plus 4 signal planes using a 2 land per channel design.

There are 6 I/O slots on the IBM 6151 system board.

- Five with a 62-pin and a 40-pin board-edge slot
- One with a 62-pin board-edge slot.

Slot	Unique	Connectors	Assignment
1	No	62-Pin	Option
2	No	62-Pin/40-Pin	Option
3	No	62-Pin/40-Pin	Option
4	No	62-Pin/40-Pin	Option
5	Yes	62-Pin/40-Pin	Coprocessor/Option
6	No	62-Pin/40-Pin	Disk/Diskette Adapter

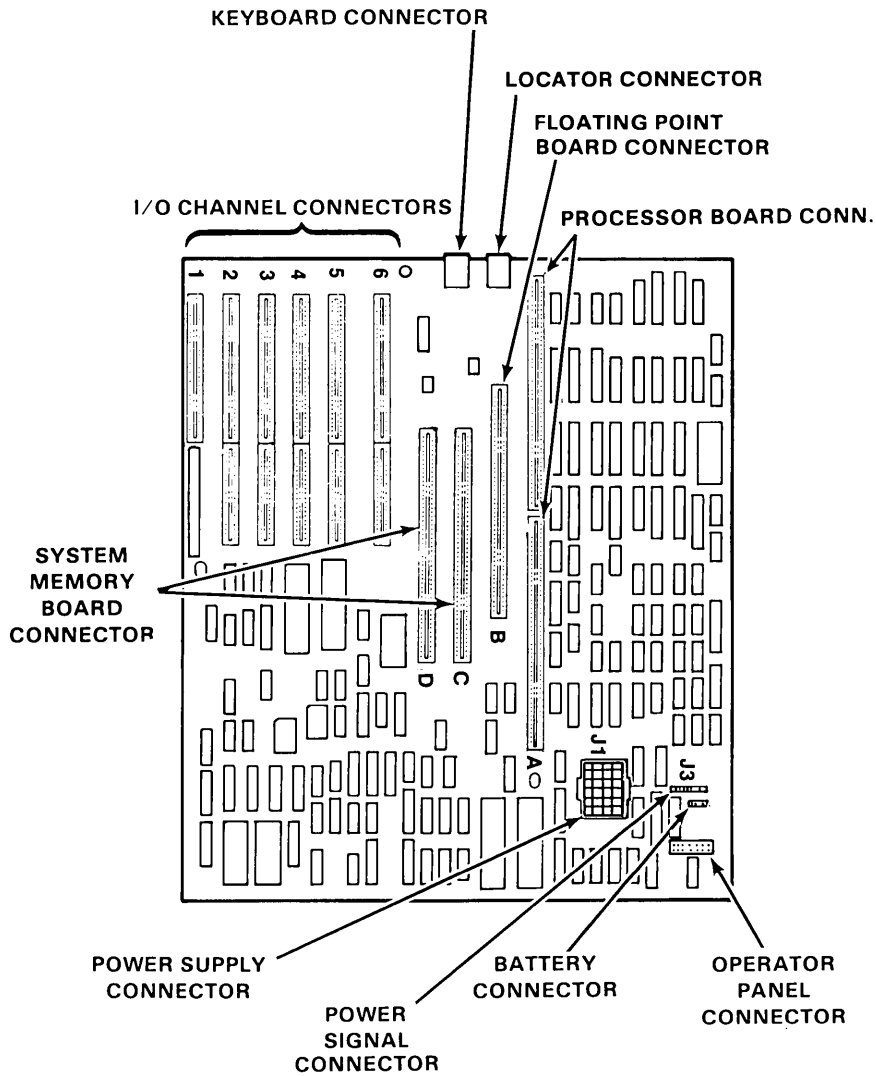


Figure 5-2. IBM 6151 Connector Locations

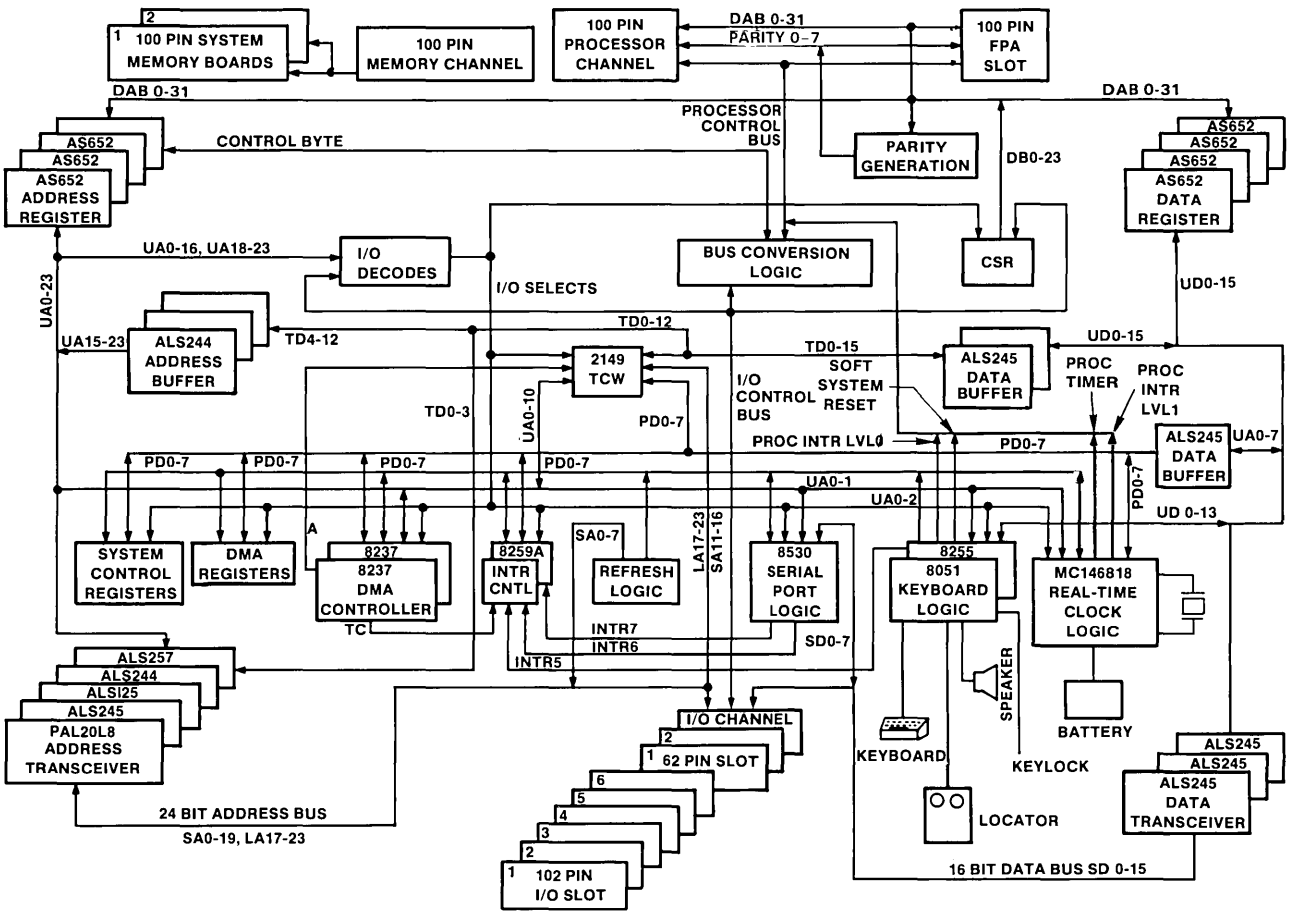


Figure 5-3. RT PC Address/Data Flow Diagram

System Board I/O Address Assignments

All the control registers, real time clock, keyboard adapter, and TCWs on the system board are accessed by load and store instructions from the system processor. Each of these facilities is assigned a unique address in I/O channel I/O address space as shown in Figure 5-4. Notice that all these facilities exist in segment 'F' in the system address map.

I/O Channel Address Bit 2 3	1 1 6 5	8 7	0	Facility Name	Facility Size	Access Read/ Write	System Hex Address
0000 00x0	1000	0000	0000 00rr	Z8530 (Serial Port)	8 Bit	R/W	F0008000-F0008003
0000 00x0	1000	0000	0010 00xx	CH A Ext Reg	8 Bit	R/W	F0008020
0000 00x0	1000	0000	0100 00xx	CH B Ext Reg	8 Bit	R/W	F0008040
0000 00x0	1000	0000	0110 00xx	8530 Intack	8 Bit	R	F0008060
0000 00x0	1000	0000	1110 00xx	I/O Delay Reg(IDR)	8 Bit	W	F00080E0
0000 00x0	1000	0100	0xxx xrrr	Keyboard Adapter	8/16 Bit	R/W	F008400-F0008407
0000 00x0	1000	1000	00rr rrrr	RTC	8 Bit	R/W	F0008800-F000883F
0000 00x0	1000	1000	010x rrrr	8237 #1	8 Bit	R/W	F0008840-F000884F
0000 00x0	1000	1000	011r rrrx	8237 #2	8 Bit	R/W	F0008860-F000887F
0000 00x0	1000	1000	100x xxxr	8259 #1	8 Bit	R/W	F0008880-F0008881
0000 00x0	1000	1000	101x xxxr	8259 #2	8 Bit	R/W	F00088A0-F00088A1
0000 00x0	1000	1000	110x xxxx	DMA Reg - DBR	8 Bit	R/W	F00088C0
0000 00x0	1000	1000	111x xxxx	DMA Reg - DMR	8 Bit	R/W	F00088E0
0000 00x0	1000	1100	0000 00xx	CH 8 Enable Reg	8 Bit	R/W	F0008C00
0000 00x0	1000	1100	0010 00xx	Control Reg - CCR	8 Bit	R/W	F0008C20
0000 00x0	1000	1100	0100 00xx	CRRA Reg	8 Bit	R/W	F0008C40

Figure 5-4 (Part 1 of 2). System Board I/O Address Map

I/O Channel Address Bit 2 3	1 1 6 5	8 7	0	Facility Name	Facility Size	Access Read/ Write	System Hex Address
0000 00x0	1000	1100	0110 00xx	CRRB Reg	8 Bit	R/W	F0008C60
0000 00x0	1000	1100	1000 00xx	Memory Config Reg	8 Bit	R	F0008C80
0000 00x0	1000	1100	1010 00xx	DIA Reg	8 Bit	W	F0008CA0
0000 00x1	0000	0rrr	rrrr rrr0	TCW	16 Bit	R/W	F0010000-F00107FE
0000 00x1	0000	1xxx	xxxx xxxx	CSR	32 Bit	R/W	F0010800

Figure 5-4 (Part 2 of 2). System Board I/O Address Map

Note: x = don't care condition
r = an address line decoded by the corresponding facility

Addressing Modes

The RT PC system board is capable of responding to two different sources of addresses. The first of these is from the processor channel. Most accesses are from this source, but the system board is also able to respond to accesses from alternate controllers residing on the I/O channel. In most cases, access from an alternate controller is destined for system memory and is passed on to the processor channel, but operations are allowed to registers on the system board under some circumstances.

System Processor Accesses

System processor accesses to system board registers or the I/O channel pass through the IOCC (Input/Output Channel Converter) on the system board. Portions of the IOCC monitor this process and sometimes change the progress of the transfer. The following is a description of how the system board handles accesses from the processor channel.

Real and Virtual Modes

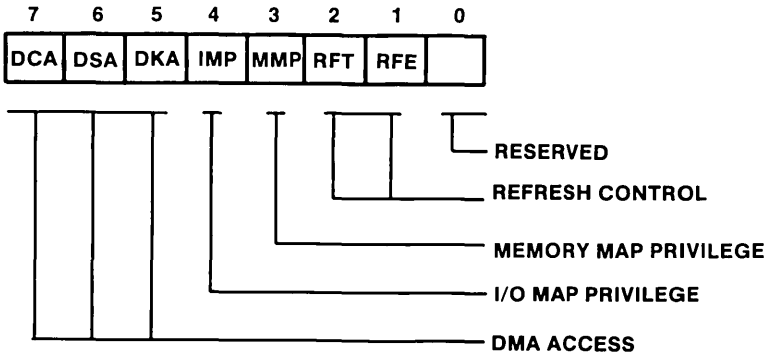
The system board makes no address distinction between the real and virtual operating modes of the system. Although slight differences occur in the way the data is transferred, neither the data itself nor the address will be treated differently whether operating in real or virtual mode.

Privileged and Unprivileged States

The system processor may be operated in privileged or unprivileged states. Although the system board does not change addressing based on the privileged state, it may be programmed to operate in a mode that differentiates between these states for access protection purposes. Privileged state is the higher of the two privilege levels. The channel control register on the system board controls access to adapters on the system board itself and on the I/O channel.

I/O and Memory Map Access

The channel control register (CCR) is part of the system board's access authorization mechanism. CCR bits 3 and 4 control accesses from the processor channel to the I/O channel and other registers on the system board. These two bits independently control access to the 64K-bytes of I/O address space and the 16M-bytes of memory address space on the I/O channel. These same bits also determine the minimum privilege level a program must be operating at to allow the transfer to take place. If an access violates this minimum privilege requirement, the transfer is not allowed and a protection violation error is reported.



Privilege IMP	Privilege MMP	Access States Allowed (Privileged/Unprivileged)
0	X	I/O map accessible only in privileged state
1	X	I/O map accessible from either state
X	0	Memory map accessible only in privileged state
X	1	Memory map accessible from either state
X = Don't care condition		

Figure 5-5. Channel Control Register (Address X'008C20')

DMA Addressing Modes

Two types of DMA (Direct Memory Access) are used within the system. They are DMA devices and alternate controllers. Refer to "I/O Channel Data Transfer" on page 6-15 for a complete description of DMA and its modes of operation.

The IOCC supports both types of DMA and provides several mechanisms which aid each type in a unique way.

Translation

The function of the translation mechanism of the IOCC is to:

- Add the appropriate number of additional high order address bits to the address supplied by either DMA device or alternate controller to form the required 32-bit address.
- Relocate within system memory selected memory references by DMA controllers on the I/O channel.
- Provide the IOCC control logic with knowledge of whether the memory reference is directed to system memory or I/O channel attached memory.

The translation of addresses within the IOCC is performed under the control of a block of control words residing within the channel. These translation control words (TCWs) contain both address translation data and control information. The translation control mechanism supports a maximum of eight DMA channels with 64 TCW entries per channel in page mode and 512 TCW entries (shared by all channels) in region mode. The TCWs are at address X'010000' through X'0107FE'.

Mode Combinations					
System/ Alt Cntl	**Page/ Region**	**Real/ Virtual**	**Valid**	**Error Detected**	**Comments**
System	Page	Real	Yes		
System	Page	Virtual	No	No	Uses Real Mode
System	Region	Real	No	Yes	Invalid DMA Operation
System	Region	Virtual	No	Yes	Invalid DMA Operation
Alt Cntl	Page	Real	Yes		
Alt Cntl	Page	Virtual	No	No	Uses Real Mode
Alt Cntl	Region	Real	Yes		
Alt Cntl	Region	Virtual	Yes		

Figure 5-6. DMA Addressing Mode Combinations

Translation Modes

The translation control word (TCW) mechanism supports two distinct translation modes, page and region. Each of these two DMA modes are tailored for a particular type of operation.

Page mode is used primarily for DMA devices using the system DMA controller. These devices are usually block structured and transfer data relatively infrequently. The diskette adapter is an example of these devices. The maximum transfer size for page mode devices is 64K-bytes (16-bit devices are limited to 128K-bytes).

Region mode is used only by alternate controllers which need to address or transfer more than 64K-bytes at a time. Region mode allows the adapter to transfer data anywhere within the 16M-byte address space. Region mode also allows the adapter to perform data transfers to adapters in the 64K-byte I/O address space.

Page Mode

In page mode each TCW entry maps a 2K-byte memory area on the I/O channel to a 2K-byte page in the system memory real map. This mode uses the upper 128K-bytes of addressability in the memory map of the I/O channel as a window into system memory. Alternate controllers accessing this area have their access translated to the appropriate address. System DMA controller accesses are treated as if all accesses involve the upper 128K-bytes, thus forming a full 24-bit address from the 16-bits output by the system DMA controller (See Figure 5-7 on page 5-18). This mode can be used by either the system DMA controller or by alternate controllers. A control bit of the selected TCW entry permits the system DMA controller to target the operation for either I/O channel attached memory or system memory. Thus, an address originally targeted somewhere in the upper 128K-byte area of the 16M-byte address map may be translated to anywhere in the 16M-byte address range either of system memory or I/O channel attached RAM. (The IOCC will not invoke the translation facility for a transfer by an alternate controller to the I/O channel).

Region Mode

In region mode each TCW entry maps a 32K-byte region on the I/O channel to a 32K-byte region of system memory. The access may be either real or virtual, depending on a control bit within the TCW entry. In this mode the complete 16M-byte I/O channel memory map can be selectively mapped onto either the real or virtual map of system memory. This mode is used only by alternate controllers. Attempts to use region mode by DMA devices operating under system DMA controllers will be reported as "Invalid DMA Operations".

Figure 5-6 on page 5-16 lists all mode combinations and shows which are valid from a hardware point of view. The two entries dealing with virtual page mode are not supported but are not detected or reported as errors. The transfer takes place but data is directed to memory segment '00XXXXXX' instead of 'E0XXXXXX'.

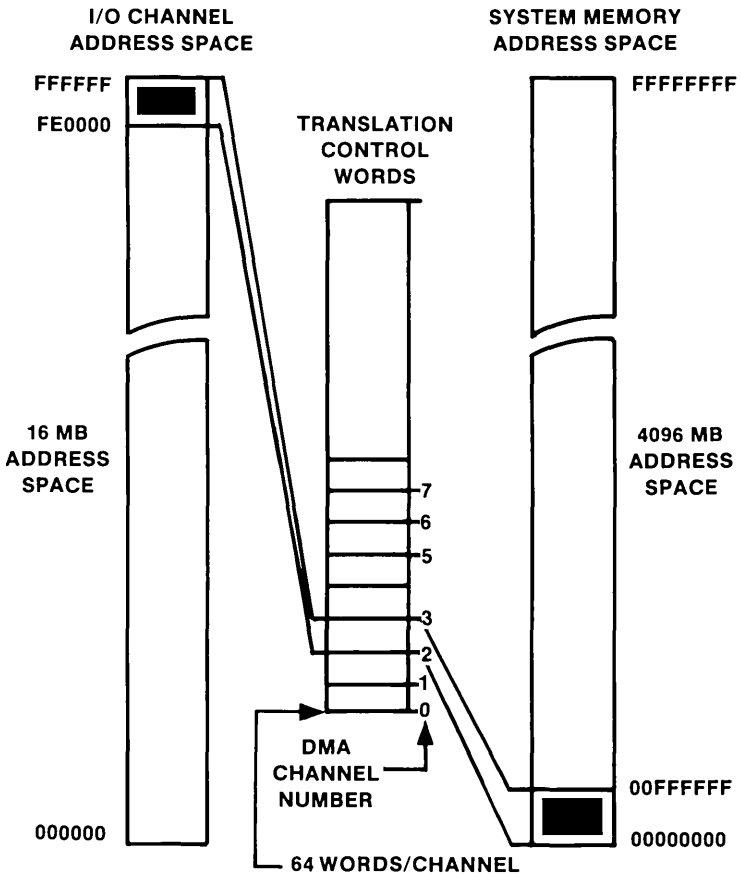


Figure 5-7. Address Translation (Page Mode) Real Address Format

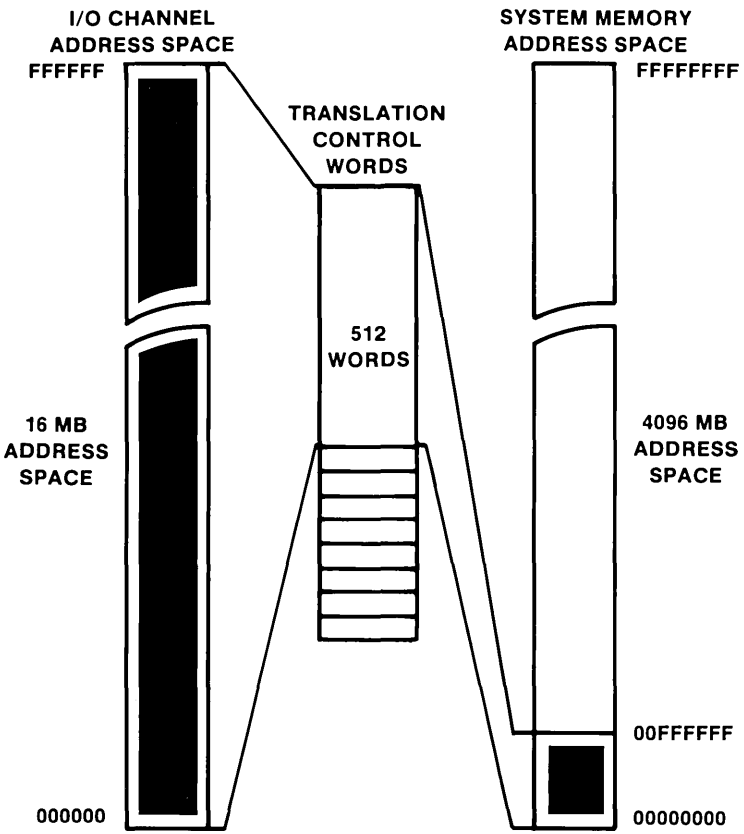


Figure 5-8. Address Translation (Region Mode) Real Address Format

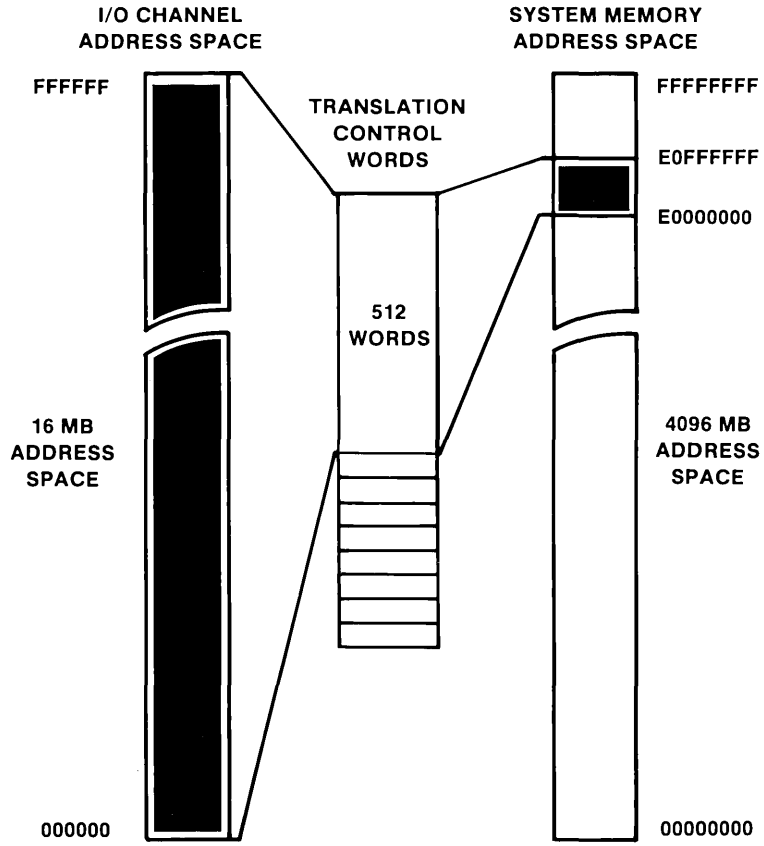
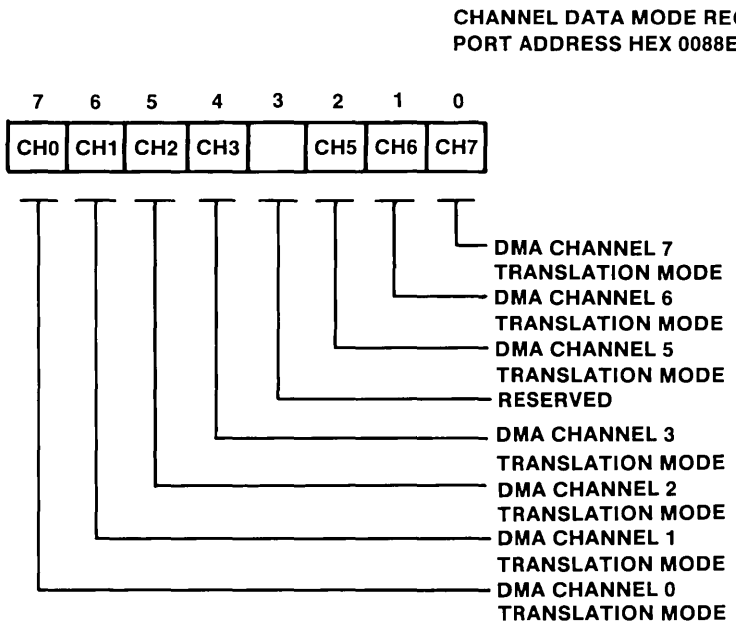


Figure 5-9. Address Translation (Region Mode) Virtual Address Format

Selection of page or region mode is under control of the DMA Mode Register (DMR). This register permits mode control for all channels to be under program control. DMA channel 8 operates only in region mode.



DMA Mode Bit	DMA Translation Mode Used
0	Page mode (2K-byte blocks)
1	Region mode (32K-byte blocks)

Figure 5-10. DMA Mode Register (System Address X'F00088E0')

Translated Address Format

The memory address width on the I/O channel is 24 bits while the memory address width at the processor channel is 32 bits. The address supplied by the system DMA controller is 16 bits and the address supplied by an alternate controller is 24 bits. The IOCC mechanism forms the appropriate address length from the address supplied by the adapter (16 or 24 bits). This process involves a combination of prefix concatenation and overlays. The translation process is applied to all addresses received from an adapter (16- or 24-bits).

Real and Virtual Modes

The IOCC supports real and virtual memory accesses to system memory. These modes differ in the manner in which the address is formed and in control signals activated during the transfer of the address to the processor channel.

Real

The real address format is used by both system DMA controllers and alternate controllers to form a 24-bit address when accessing system memory in real mode. For alternate controllers using region mode, the IOCC uses the low order 15 bits received from the adapter and replaces the upper 9 bits with a translated value. For either alternate controllers or system DMA controller using page mode, the IOCC uses the low order 11 address bits supplied and concatenates a 13-bit prefix to form a 24-bit address. In either case, the resulting address is sent to system memory with the Memory Management translate function turned off.

Virtual

The virtual address format is used to form a 32-bit address when accessing system memory in virtual mode. This addressing format can be used only by alternate controllers operating in region mode. The IOCC forms a 32-bit address by concatenating a 17-bit prefix onto the low order 15-bits of address supplied by the adapter. The resulting address is sent to system memory with the memory management unit translate function bit turned on.

Figure 5-11 on page 5-23 through Figure 5-14 on page 5-26 show how the translation mechanism of the IOCC bridges the gap between the DMA source address on the I/O channel and the resulting target address sent to the processor channel for each addressing mode and address source.

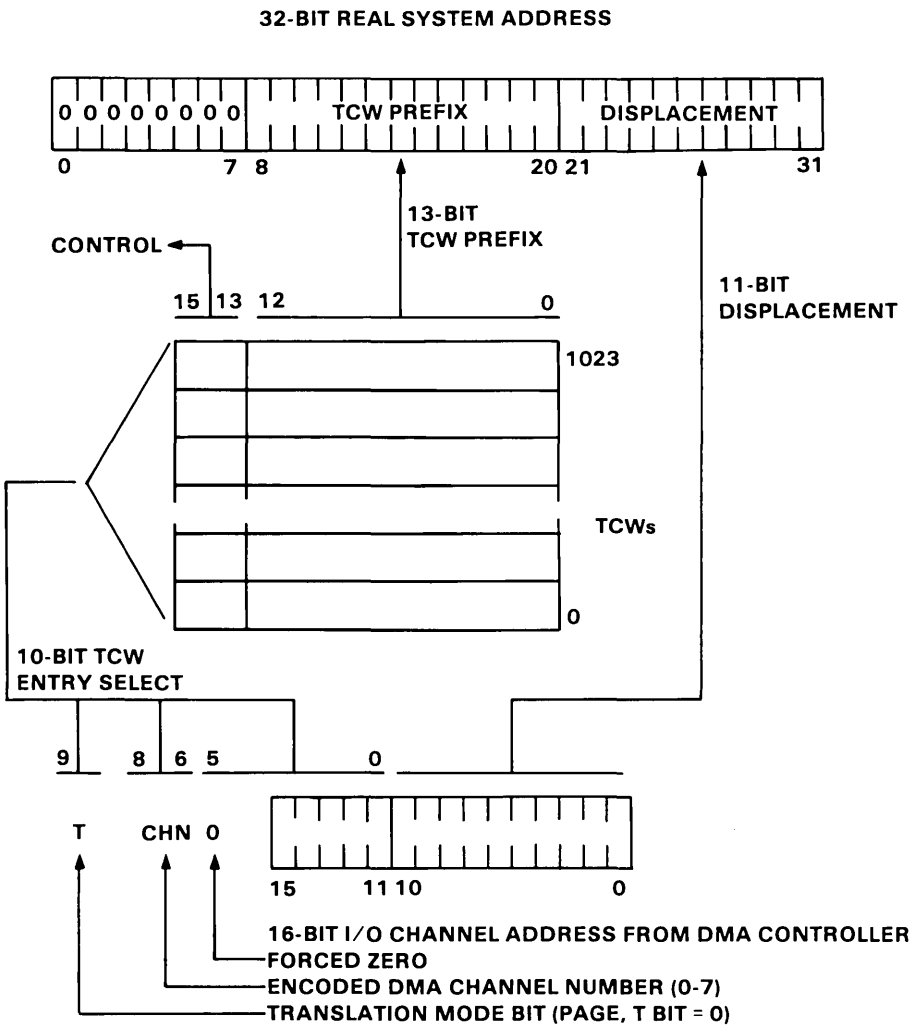


Figure 5-11. System DMA - Page Mode (8-Bit Channel) Translated Address Formation

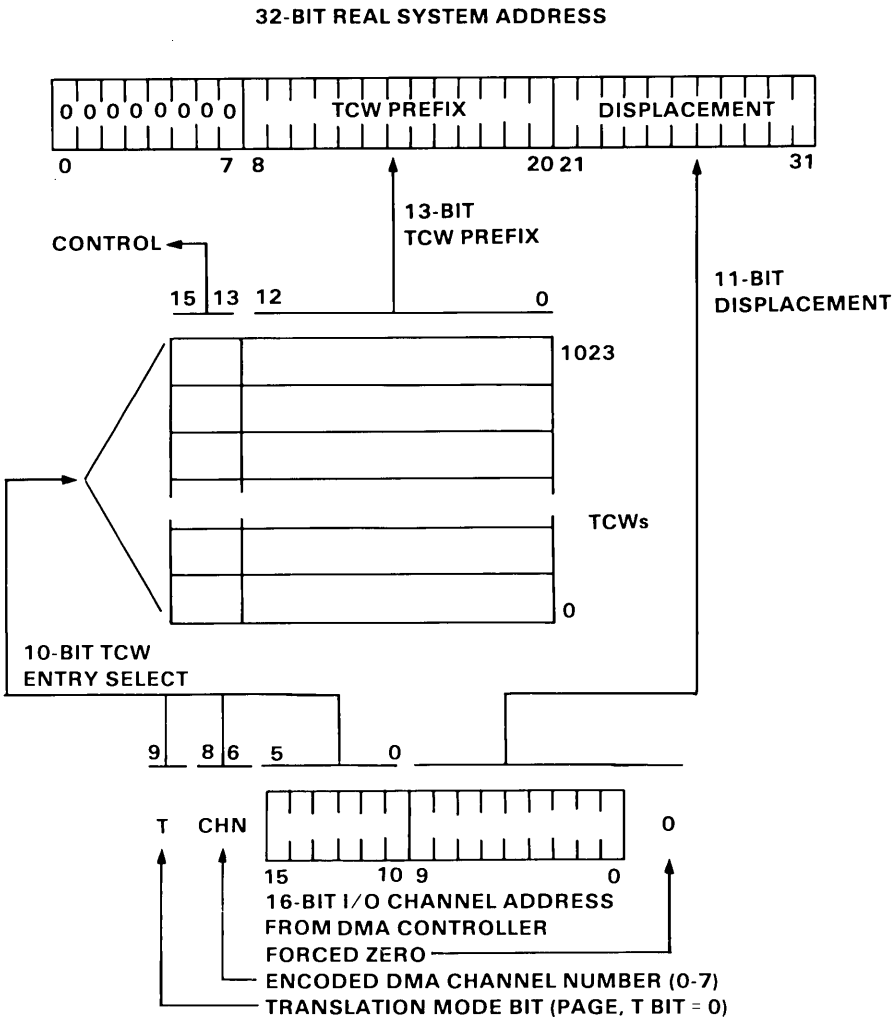


Figure 5-12. System DMA - Page Mode (16-Bit Channel) Translated Address Formation

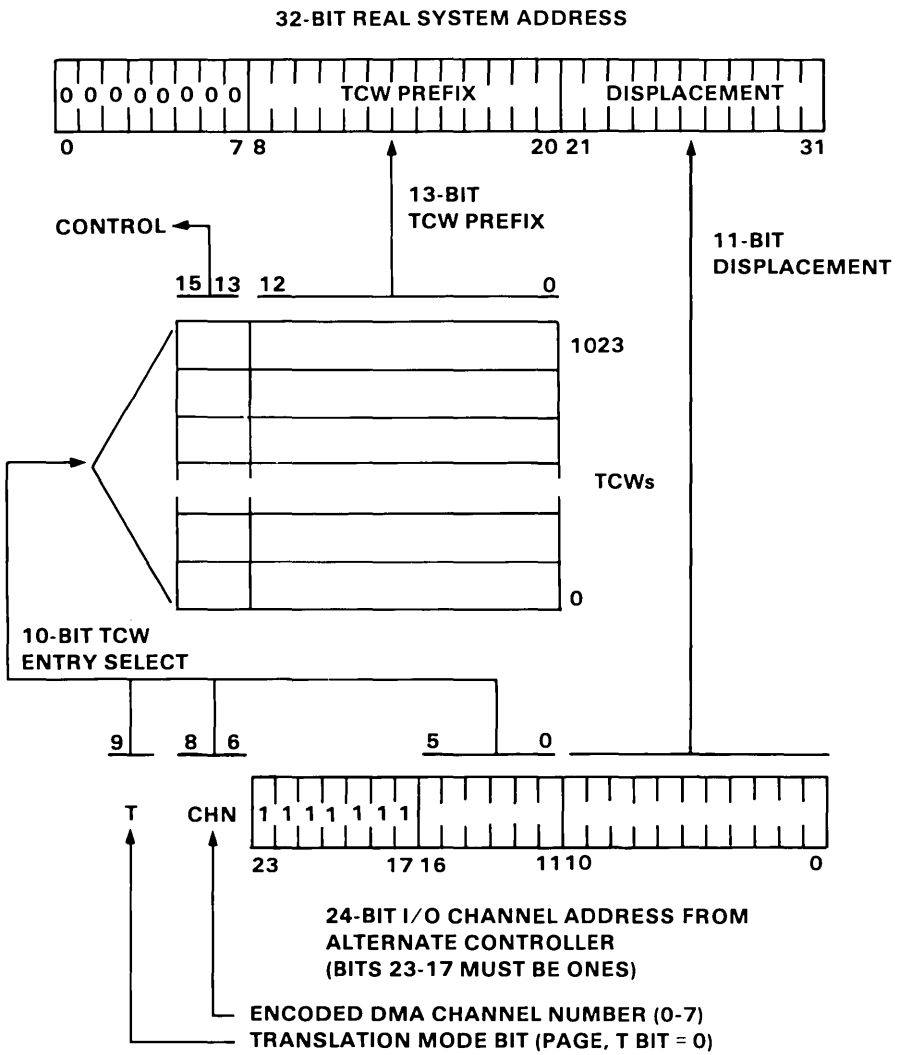


Figure 5-13. Alternate Controller - Page Mode Translated Address Formation

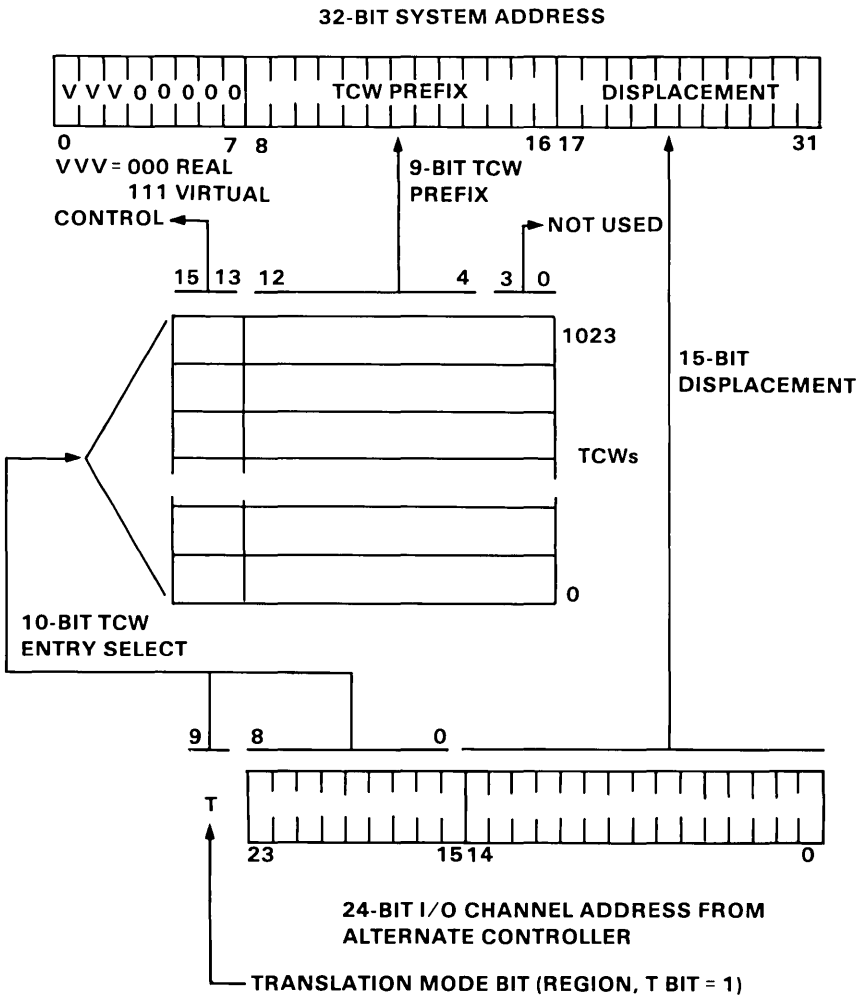


Figure 5-14. Alternate Controller-Region Mode Translated Address Formation

Translation Control Words

The translation control word (TCW) entries associated with a given DMA adapter must cover the complete address range which that adapter is capable of addressing. The address range covered by one TCW entry is dependent on the mode setting for that channel. In region mode each TCW entry covers 32K-bytes while in page mode each TCW entry covers 2K-bytes.

Start Hex Address	End Hex Address	Used by Channel	Control Word Usage
010000	01007E	Channel 0	Page mode
010080	0100FE	Channel 1	Page mode
010100	01017E	Channel 2	Page mode
010180	0101FE	Channel 3	Page mode
010200	01027E	Reserved	Reserved
010280	0102FE	Channel 5	Page mode
010300	01037E	Channel 6	Page mode
010380	0103FE	Channel 7	Page mode
010400	0107FE	All channels	Region mode

Figure 5-15. Translation Control Word (TCW) Port Addresses

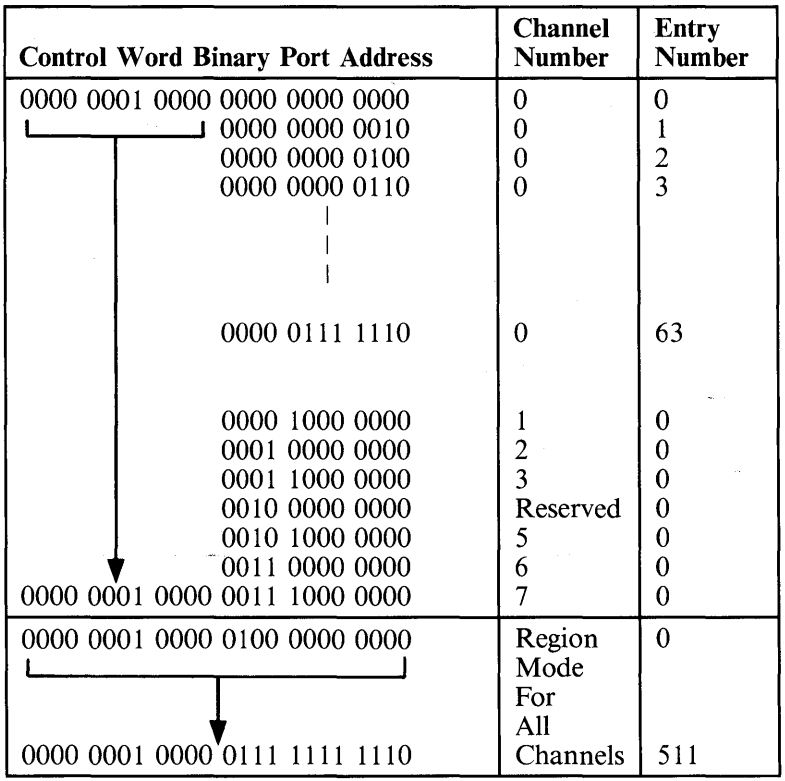


Figure 5-16. Translation Control Word (TCW) Binary Port Address

Each part of the TCW is 16 bits wide. Figure 5-17 on page 5-29 shows the format of the TCW. Unlike other 16-bit registers, the TCW words are the same as the processor register image. The high and low order bytes of the word are not reversed as they pass through the IOCC.

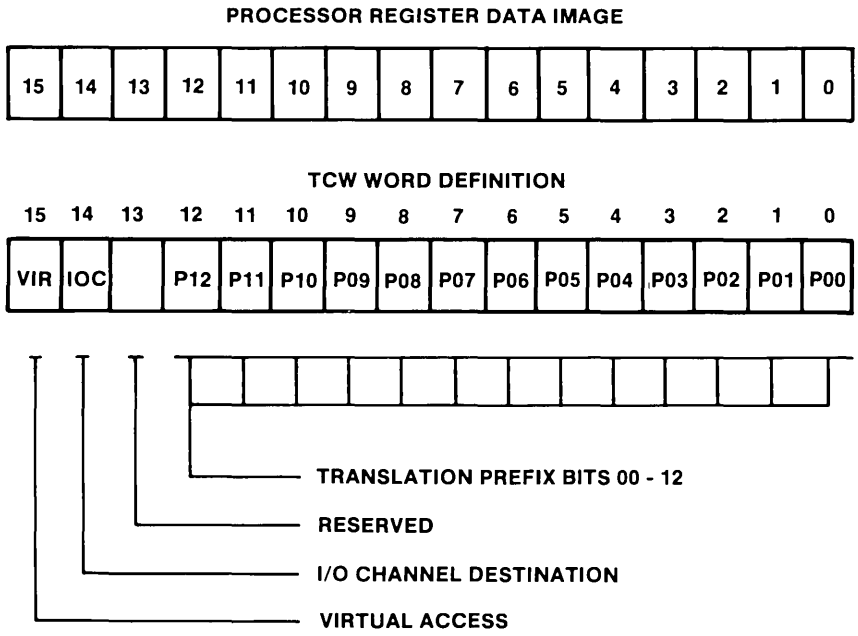


Figure 5-17. TCW Format

Access Control

This field consists of the virtual access and I/O channel destination bits. It permits the software to select the target memory unit (system memory or I/O channel attached) which is used in the current operation and the memory management unit transfer type. Bit 14 (IOC) defines system memory or I/O channel access while bit 15 (VIR) differentiates between real or virtual address formats. The definition of this field is as follows:

Control Bits		Resulting Action	
VIR	IOC	System DMA Controller	Alternate Controller
0	0	Translate to real address format and send to system memory.	Translate to real address format and send to system memory.
1[1]	0	Translate to real address format and send to system memory.	Translate to virtual address format and send to system memory.
X	1	Translate to real address format and send to I/O channel.	No action by IOCC alternate controller, uses unchanged address.

X = Don't care condition
[1] System DMA controller ignores the virtual bit state.

Figure 5-18. Virtual Access and I/O Channel Destination Bits

DMA Access Authorization

The Channel Control Register (CCR) controls part of the system board's access protection mechanism. Three CCR bits (5, 6, 7) control accesses from alternate controllers to certain registers and adapters on the system board. Access to adapters or memory located on the I/O channel by alternate controllers cannot be prevented.

Alternate controller access to system memory is controlled by the translation mechanism. (Alternate controllers cannot access the translation control word or the channel status register.)

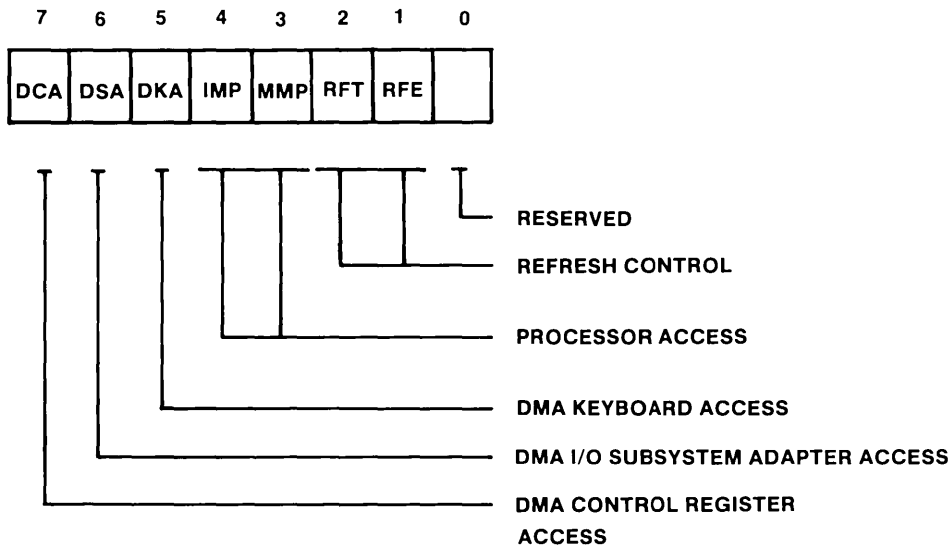


Figure 5-19. Channel Control Register (Address X'008C20')

DMA Access DCA	DSA	DKA	Alternate Controller Access Allowed or Denied	Affected Address Range
0	X	X	Allow access to control registers	008CY0
1	X	X	Do not allow access to control registers	008CY0
0	0	X	Allow access to I/O subsystem adapters	0088YY
0	1	X	Do not allow access to I/O subsystem adapters	0088YY
0	X	0	Allow access to keyboard adapter	00840Y
0	X	1	Do not allow access to keyboard adapter	00840Y

X = Don't care condition
Y = Several possible addresses

Figure 5-20. DMA Access Bits

Arbitration

System arbitration is handled by the system arbiter which resides on the system board. It is a custom design for the RT PC system. The system arbiter is aided by the arbitration mechanism of two 8237 DMA controllers for DMA operations. The system arbiter function sets priorities and grants requests for using the I/O channel. The following is a list of entities capable of requesting I/O channel usage. They are listed in descending priority.

- 8237 DMA controller 1
- Refresh for I/O channel dynamic RAM
- Processor Input/Output (PIO)
- 8237 DMA controller 2
- Coprocessor.

System Arbiter Operation

The processor board is the default owner of the system arbiter. The processor board carries out read and write operations to the system board registers and the I/O channel while the system arbiter is reset. The system arbiter must be enabled to arbitrate and grant requests to use the I/O channel. Refresh (refer to "Refresh" on page 5-53) and the 8237's (refer to "DMA" on page 5-39) must be enabled by system software before they can request the use of the I/O channel from the system arbiter. When the system arbiter receives a request to use the I/O channel, it must first be granted control of the I/O channel by the processor board or be in control of the I/O channel. After the processor board grants control of the I/O channel to the system arbiter, the system arbiter grants control of the I/O channel to the requestor.

The linear arbitration priority scheme is; 8237 DMA controller 1, Refresh, PIO, 8237 DMA controller 2 and Coprocessor. The highest priority device requesting the channel is granted control when the channel is released by another device. When a device gains control of the channel it cannot be preempted (except for Coprocessor). Devices should not keep control of the channel for more than 8 microseconds to prevent locking out other devices.

The Coprocessor has the capability of using the I/O channel for extended amounts of time. Once the Coprocessor receives control of the I/O channel, it can keep control until it is finished or until the arbiter requests the Coprocessor to relinquish the channel by removing its acknowledge. The acknowledge to the Coprocessor is never taken away until after the Coprocessor has activated the '-Master' signal on the I/O channel. See "I/O Slot Uniqueness" on page 6-54 and "Coprocessor Arbitration" on page 6-56 for more information.

When the system arbiter is disabled it will allow PIOs and only make a request to the processor board for use of the I/O channel for refresh. All DMA requests to the system arbiter are ignored

when it is disabled. The arbiter is placed in the disabled mode by writing to the channel reset register B (CRRB). See "Component Reset Register B (CRRB)" on page 5-61

The system arbiter is disabled when the memory management unit sends an exception reply on a DMA transfer. This forces a reset to the system DMA controllers and causes the 8237's to relinquish the I/O channel to allow software the opportunity to rectify the problem if the exception was caused by the 8237. In the event the exception was caused by the Coprocessor, the Coprocessor acknowledge is deactivated. The Coprocessor should get off the I/O channel as soon as possible.

The system arbiter is also disabled when the 'POR' signal goes active or when the processor board resets the IOCC.

The arbitration mechanism of the 8237 DMA controller aids the system arbiter. Individual DMA requests are monitored and priorities set by the 8237's. When the 8237 receives an acknowledgment from the system arbiter, the 8237 determines which 'DACK' to grant. Activating the 'DACK' line grants the I/O channel to the requestor.

Single Cycle

Single cycle operations on the I/O channel are the standard method for data transfer. This insures that correct system operation is maintained and that all adapters and options have timely access to the I/O channel.

Burst Cycles

Burst cycles are supported and achieved by an alternate controller keeping the DMA request active after gaining control of the I/O channel (DRQ active). Burst mode can be used for transfers to either system memory or to I/O channel attached memory. An alternate controller must not hold the I/O channel for more than 8 microseconds or correct system operation cannot be guaranteed. When a DMA adapter starts its DMA cycle, (or cycles) it does not know how many microseconds are left until another requestor requires the I/O channel.

Buffer Mode DMA

Buffer mode DMA is achieved by an alternate controller keeping the DMA request active after gaining control of the I/O channel (DRQ active) for pairs of I/O cycles. Buffer mode DMA allows alternate controllers to transfer 4 bytes to system memory in a single 4-byte DMA request cycle to system memory only. This is achieved by the appropriate DMA channel bit being enabled in the DMA buffer register. See "Buffer Register" on page 5-54. The IOCC uses this data buffer

register bit to generate the correct responses to the I/O channel and the processor channel for buffer mode DMA transfers.

This I/O channel feature reduces the processor channel traffic. In buffer mode DMA, the alternate controller does two 2-byte I/O channel cycles to or from the IOCC for each 4-byte wide cycle on the processor channel or with system memory. Buffer mode DMA operations can be real or virtual transfers and must be 4 bytes wide. All buffer mode DMA operations must start on double word boundaries with SA0=0 and SA1=0. The first I/O channel cycle is done with SA0=0 and SA1=0. The next I/O channel cycle is done at the next higher word address SA0=0 and SA1=1.

Buffer mode DMA reads are initiated the same as single cycle DMA reads. In buffered mode the IOCC initiates a 4-byte system memory read. 'I/O CH RDY' will go not ready until the data is available. When the data is available the alternate controller reads the first 2 bytes at the double word boundary. The alternate controller increments the address by one word and the IOCC multiplexes the second 2 bytes to the I/O channel where they are read by the alternate controller. This completes a buffer mode DMA read. The alternate controller must now conform to the DMA timings to get off the I/O channel.

Notes:

1. On the first read cycle, 'I/O CH RDY' not ready must be used correctly. On the second read cycle, the alternate controller must conform to all appropriate default timings, address valid, memory access and address hold times for correct operation.
2. Reading from an address with SA0 = 0 and SA1 = 1 for the first address of a coupled pair just before reading the same address with SA0 = 0 and SA1 = 0 causes the alternate controller to receive bad data with no error condition generated. Reads to an address with SA0 = 1 results in an interrupt to the processor board. The DMA error bit is set in the channel status register (CSR). The error is posted in the error log but is not reported directly to the user.

Buffer mode DMA writes are initiated the same way as single cycle DMA writes. In buffered mode, the alternate controller writes the first two bytes of data to an address at a double word boundary. The IOCC latches the address on the I/O channel address bus (LA and SA bits), and latch the 2-bytes of data in the IOCC when '-MEMW' goes negative. No data is transferred to system memory at this time. 'I/O CH RDY' will not go ready and the cycle executes at the default cycle time.

The alternate controller increases the address on the I/O channel by one word to set up the IOCC for the next write. The IOCC must multiplex the second 2 bytes to the buffer so that 4 bytes are ready for the write to system memory. This time when '-MEMW' goes negative, 'I/O CH RDY' will go *not ready* until the data is written into system memory. When 'I/O CH RDY' goes ready, the alternate controller has completed a buffer mode DMA write. The alternate controller must conform to default cycle write timings through out this cycle.

Note: Writing to an address with SA0 = 0 and SA1 = 1 for the first address of a coupled pair just before writing the same address with SA0 = 0 and SA1 = 0 causes the alternate controller to send bad data with no error condition generated. Reads to an address with SA0 = 1 results in an

interrupt to the processor board. The DMA error bit is set in the channel status register (CSR). The error is posted in the error log but is not reported directly to the user.

Burst and Buffer Mode DMA

Burst and buffer mode DMA can be used together by an alternate controller. All rules governing both burst cycles and buffered mode DMA must be strictly adhered to. The alternate controller must not hold the I/O channel for more than 8 microseconds or correct system operation cannot be guaranteed.

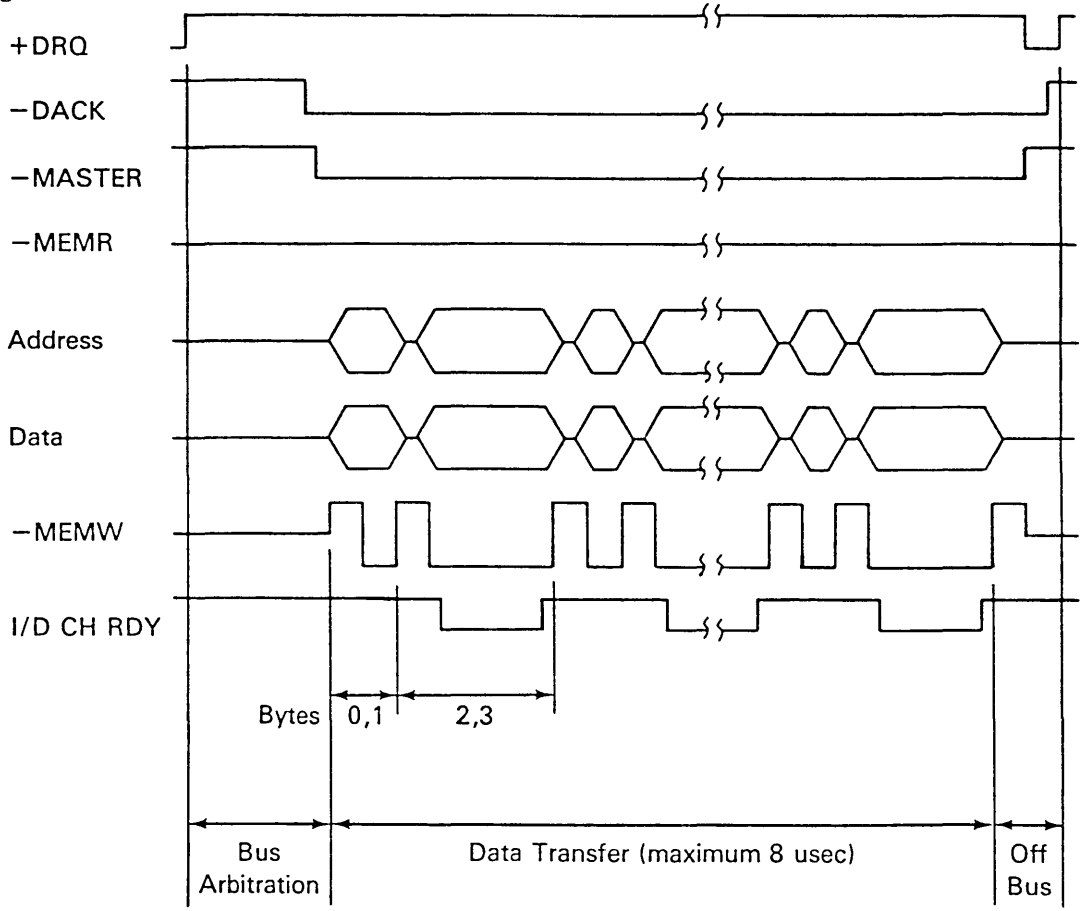


Figure 5-21. Buffered and Burst DMA Write to System Memory Timings

I/O Channel Operations

The system board accepts commands from the processor board and converts those commands to memory and I/O channel operations. For a detailed description of the I/O channel, see Section 6.

Fast and Slow Channel Operations

A fast channel operation occurs when a 16-bit memory device activates the I/O channel signal '-MEMCS16' on a memory operation or a 16-bit I/O device activates '-IOCS16' on an I/O operation. A fast channel operation results in an I/O channel cycle with the read/write control signal being active for 350 nanoseconds. Slow channel cycles occur when the I/O channel signals '-MEMCS16' and '-IOCS16' are inactive on memory or I/O operations. The read/write channel control signals for slow channel operations are active for 650 nanoseconds.

Single and Multicycle Channel Operations

The processor board can request 8, 16, and 32-bit data transfers from the I/O subsystems or the I/O channel.

- Eight-bit transfers to 8-bit adapters result in a single I/O channel cycle
- Eight-bit transfers to 16-bit adapters result in a single I/O channel cycle
- Sixteen-bit transfers to 16-bit adapters result in a single I/O channel cycle
- Sixteen-bit transfers to 8-bit adapters are converted to two 8-bit transfers by the system board.
- Thirty two-bit transfers to 8-bit devices are converted to four 8-bit transfers
- Thirty two-bit transfers to 16-bit devices are converted to two 16-bit transfers.

The system board increases I/O channel address bits SA0 and SA1 by one on 8-bit multicycle memory operations and by two on 16-bit multicycle memory operations. SAO and SA1 are not increased on multicycle transfers to devices in the I/O address space.

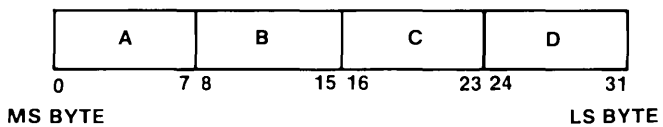


Figure 5-22. Processor Board to System Board Data Interface

ADDESS SA1	ADDESS SA0	TYPE TRANSFER	DATA BYTE TRANSFERED	I/O CHANNEL DATA PLACEMENT 15 8 7 0	
0	0	8-BIT	A	-	A
0	1	8-BIT	B	B	B
1	0	8-BIT	C	-	C
1	1	8-BIT	D	D	D
0	0	16-BIT	A B	B	A
1	0	16-BIT	C D	D	C

Figure 5-23. Single Cycle Operation Data Placement

INITIAL * SA1	INITIAL * SA0	TYPE TRANSFER	CYCLE NUMBER	MEMORY TRANSFERS SA1	MEMORY TRANSFERS SA0	I/O TRANSFERS SA1	I/O TRANSFERS SA0	DATA BYTE TRANSFERED
0	0	32-BIT	1	0	0	0	0	A B
			2	1	0	0	0	C D

I/O CHANNEL DATA PLACEMENT

15 8 7 0

B	A
D	C

* ADDRESS SUPPLIED BY PROCESSOR BOARD

Figure 5-24. Multicycle Operation to 16-Bit Device

INITIAL * SA1	INITIAL * SA0	TYPE TRANSFER	CYCLE NUMBER	MEMORY TRANSFERS SA1	MEMORY TRANSFERS SA0	I/O TRANSFERS SA1	I/O TRANSFERS SA0	DATA BYTE TRANSFERED	I/O CHANNEL DATA PLACEMENT ** 15 8	I/O CHANNEL DATA PLACEMENT ** 7 0
0	0	16-BIT	1	0	0	0	0	A	-	A
			2	0	1	0	0	B	B	B
1	0	16-BIT	1	1	0	1	0	C	-	C
			2	1	1	1	0	D	D	D
0	0	32-BIT	1	0	0	0	0	A	-	A
			2	0	1	0	0	B	B	B
			3	1	0	0	0	C	-	C
			4	1	1	0	0	D	D	D

Figure 5-25. Multicycle Operation to 8-Bit Device

Notes:

1. * Address supplied by processor board.
2. ** Data is not expected on data bits 8-15 on reads from 8-bit devices.

DMA

The RT PC system supports eight channels of Direct Memory Access (DMA). It supports 7 channels with two 8237 DMA controllers. With the aid of the translation control word mechanism, all DMA channels can transfer data throughout the 16M-byte address spaces of system memory and I/O channel attached memory.

The two 8237's must be enabled before they can be programmed. At power on time, the 8237's are disabled. The 8237 ignores commands until it is enabled. Bits 3 and 4 of the component reset register B enable and disable the 8237's. Refer to "I/O Channel Data Transfer" on page 6-15 for more information on DMA.

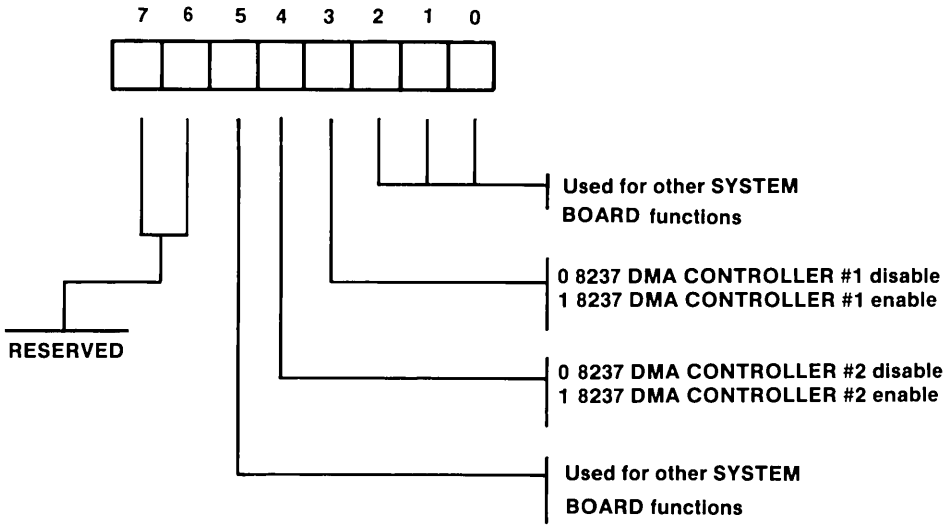


Figure 5-26. Component Reset Register B (Address X'F0008C60')

The 8237 memory to memory operation is not supported by the system board. Active high 'DACK' lines are not supported. Bits 0, 1, and 7 of the 8237's status/command register must be programmed to 0.

All internal locations, especially the mode registers, of the 8237 DMA controllers should be loaded with some value when power is turned on.

Detailed information on the use of the 8237 DMA controller can be found in the *Intel Component Data Catalog* and is not repeated here.

DMA channels 0-3 are supported by 8237 DMA controller 1. DMA channels 0-3 support 8-bit DMA devices. See "DMA Types" on page 6-16 for a description of DMA devices. DMA channels 0-3 also support 16-bit alternate controllers.

DMA channels 5-7 are supported by 8237 DMA controller 2. DMA channels 5-7 support 16-bit DMA devices and 16-bit alternate controllers. DMA channel 4 is not used.

DMA channel 8 supports only alternate controllers. DMA channel 8 is only available on the Coprocessor slot. It replaces DMA channel 7 in that slot and does not operate the same as DMA channels 0-3 and 5-7. See "I/O Slot Uniqueness" on page 6-54 and "Coprocessor Arbitration" on page 6-56 for a description of operation.

All alternate controllers must be connected to SD0 - SD15 (16-bit) and use '-SBHE'. Alternate controllers can transfer 8- or 16-bits of data. The even bytes of data must be on SD0 - SD7 and the odd bytes on SD8 - SD15.

8237 DMA Controller 1

In DMA device mode, channels 0 through 3 transfer data between 8-bit DMA devices and 8-bit and 16-bit I/O channel attached memory or system memory. Any alternate controller can use DMA channels 0 through 3. Eight-bit alternate controllers are not allowed. Refer to Figure 5-7 on page 5-18 through Figure 5-14 on page 5-26 for determination of address format for programming of the current address (base address). The current count (base word count) refers to the number of bytes to be transferred.

Address	Command
00 88 40	Channel 2 current address
00 88 41	Channel 2 current count
00 88 42	Channel 1 current address
00 88 43	Channel 1 current count
00 88 44	Channel 0 current address
00 88 45	Channel 0 current count
00 88 46	Channel 3 current address
00 88 47	Channel 3 current count
00 88 48	Status/command
00 88 49	Write request register
00 88 4A	Write mask register 1
00 88 4B	Write mode register
00 88 4C	Clear byte pointer flip/flop
00 88 4D	Read temp reg/write master clear
00 88 4E	Clear mask register (8237A-5)
00 88 4F	Write mask register 2

Figure 5-27. 8237 DMA Controller #1 Control Register Addresses

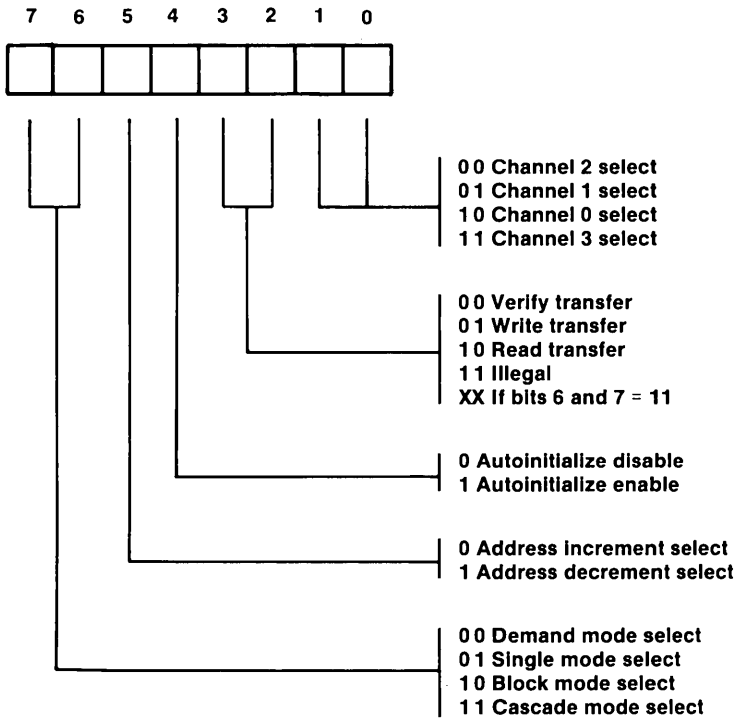


Figure 5-28. 8237 DMA Controller 1 Write Mode Register (Address X'00884B')

Note: Devices using block, cascade or command mode must not hold the I/O channel for more than 8 microseconds.

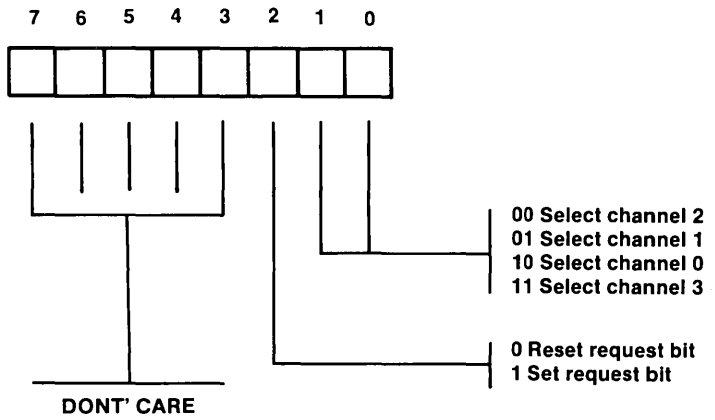


Figure 5-29. 8237 DMA Controller 1 Write Request Register(Address X'008849')

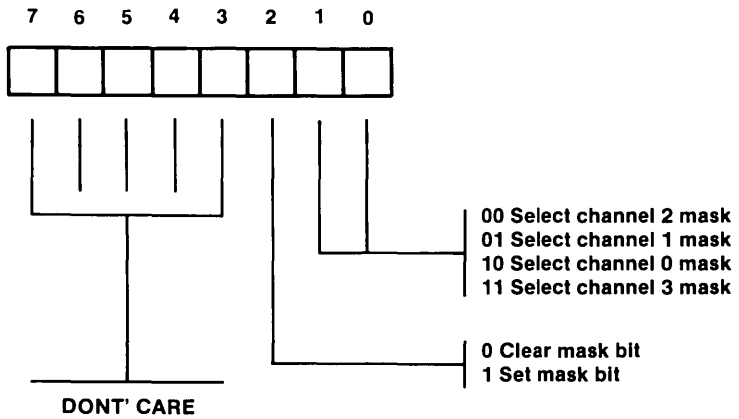


Figure 5-30. 8237 DMA Controller 1 Write Mask Register 1 (Address X'00884A')

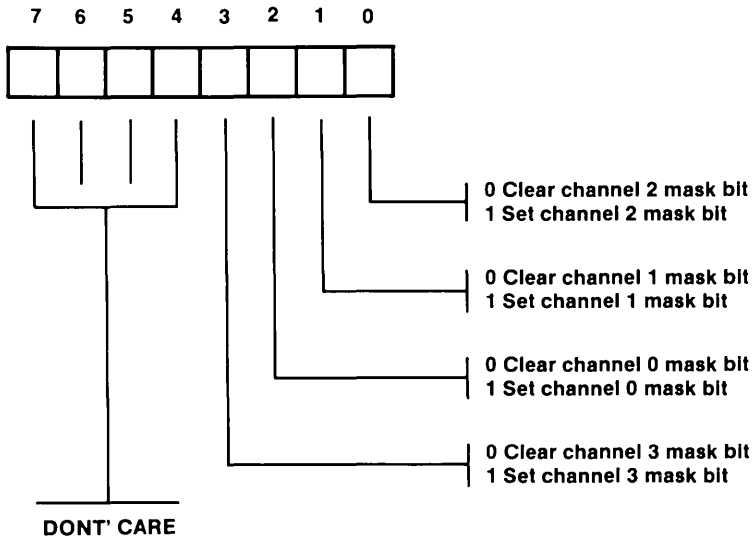


Figure 5-31. 8237 DMA Controller 1 Write Mask Register 2 (Address X'00884F')

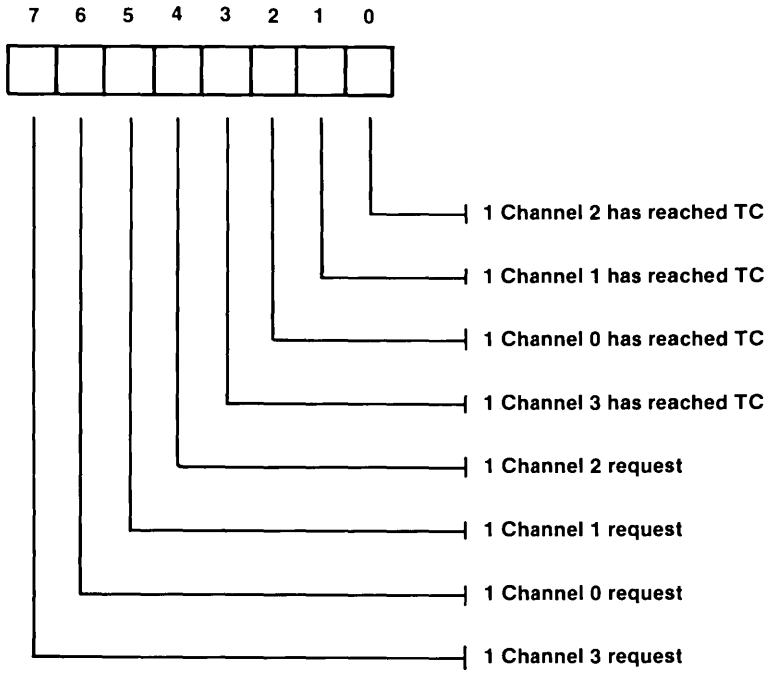


Figure 5-32. 8237 DMA Controller 1 Status Register (Address X'008848')

Note: The status/command register serves as two registers. In read mode, it is a status register. In write mode, it is a command register.

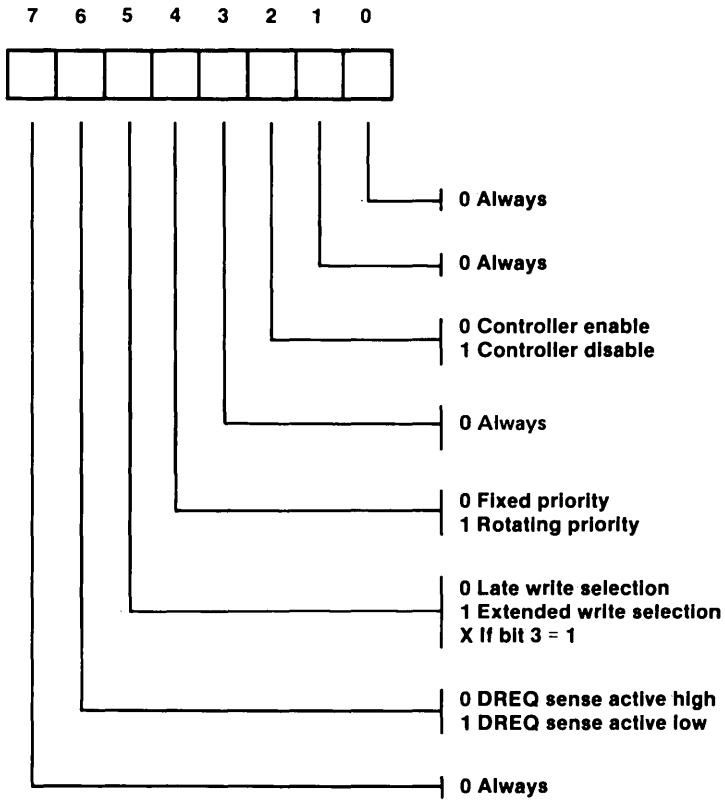


Figure 5-33. 8237 DMA Controller 1 Command Register (Address X'008848')

8237 DMA Controller 2

DMA channel 4 is not supported by the system board. Channels 5 through 7 transfer data between 16-bit DMA adapters and 16-bit I/O channel attached memory or system memory. Channels 5 through 7 can only transfer data on even boundaries. Refer to Figure 5-7 on page 5-18 through Figure 5-14 on page 5-26 for determination of address format for programming the current address (base address). The current count (base word count) refers to the number of 16-bit words to be transferred.

Address	Command
00 88 60	Channel 4 current address*
00 88 62	Channel 4 current count*
00 88 64	Channel 5 current address
00 88 66	Channel 5 current count
00 88 68	Channel 6 current address
00 88 6A	Channel 6 current count
00 88 6C	Channel 7 current address
00 88 6E	Channel 7 current count
00 88 70	Status/command
00 88 72	Write request register
00 88 74	Write mask register 1
00 88 76	Write mode register
00 88 78	Clear byte pointer flip/flop
00 88 7A	Read temp reg/write master clear
00 88 7C	Clear mask register (8237A-5)
00 88 7E	Write mask register 2

* Channel 4 not supported.

Figure 5-34. 8237 DMA Controller 2 Control Register Addresses

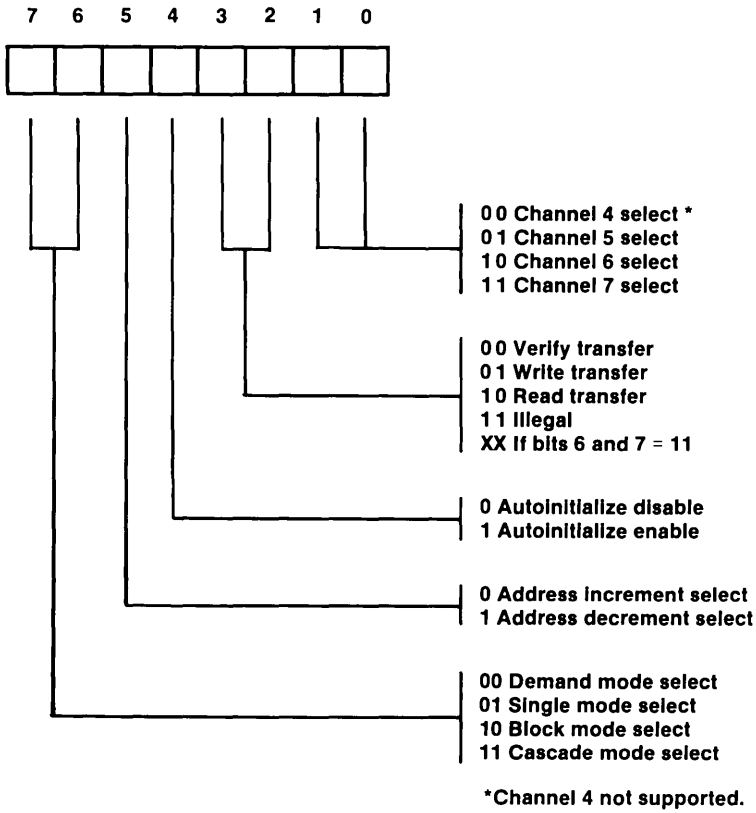


Figure 5-35. 8237 DMA Controller 2 Write Mode Register (Address X'008876')

Note: Devices using block, cascade or command mode must not hold the I/O channel for more than 8 microseconds.

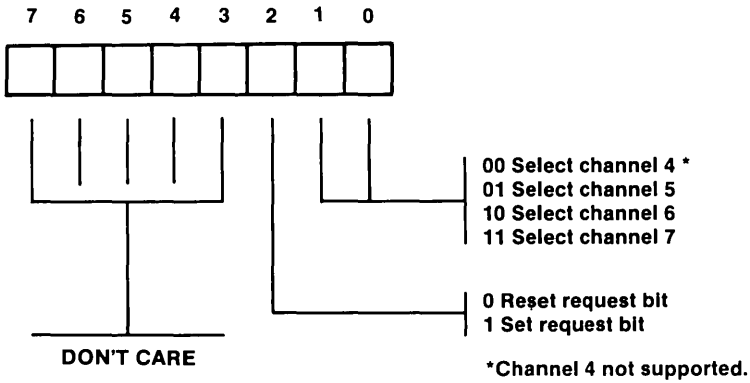


Figure 5-36. 8237 DMA Controller 2 Write Request Register (Address X'008872')

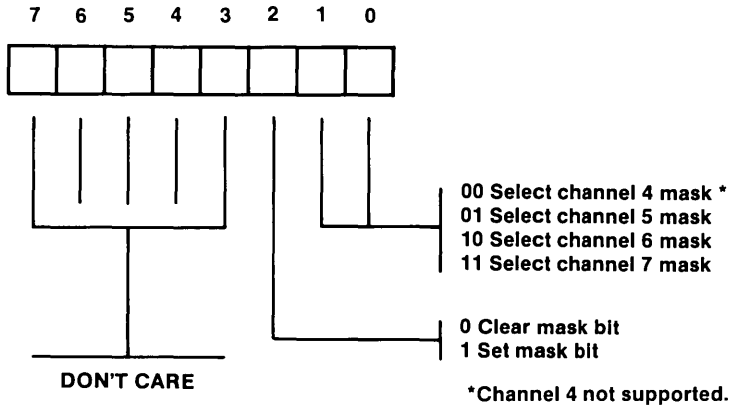


Figure 5-37. 8237 DMA Controller 2 Write Mask Register 1 (Address X'008874')

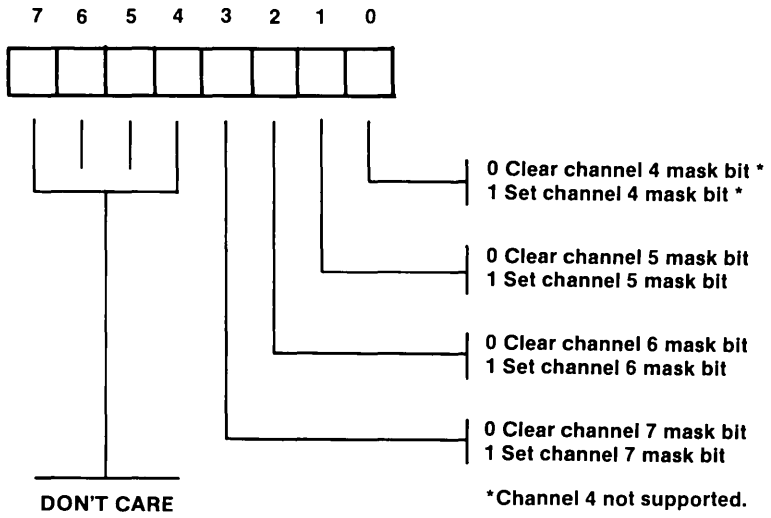


Figure 5-38. 8237 DMA Controller 2 Write Mask Register 2 (Address X'00887E')

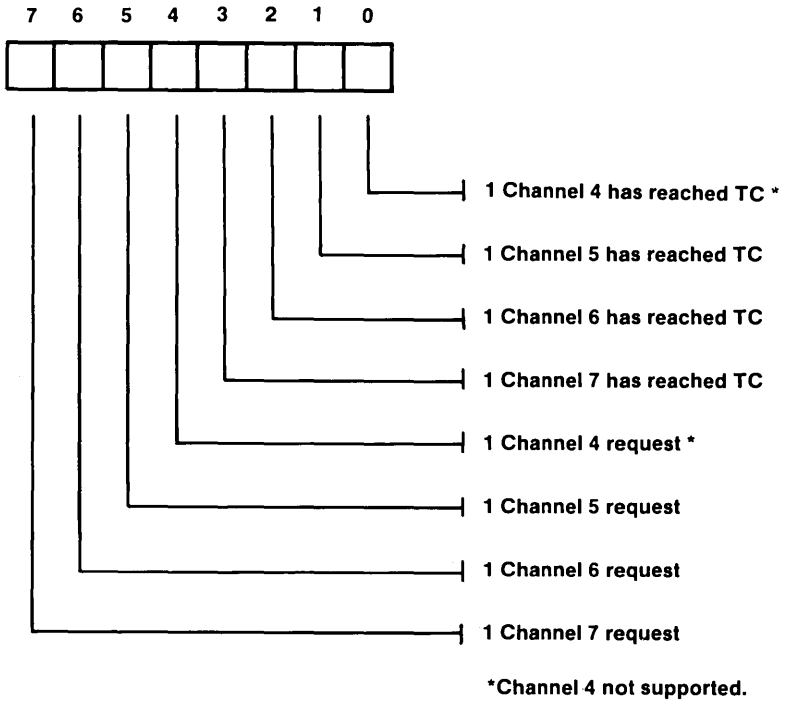


Figure 5-39. 8237 DMA Controller 2 Status Register (Address X'008870')

Note: The status/command register serves as two registers. In read mode, it is a status register. In write mode, it is a command register.

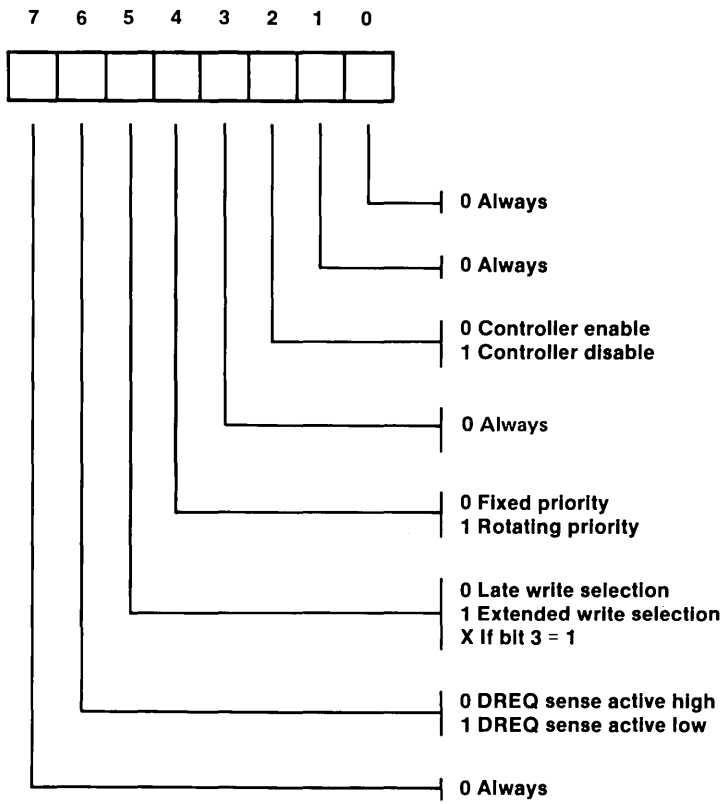


Figure 5-40. 8237 DMA Controller 2 Command Register (Address X'008870')

Alternate Controllers

The system board allows 16-bit processors or DMA controllers on the I/O channel (alternate controllers) to gain control of the I/O channel. Alternate controllers can transfer data to system memory, I/O channel attached devices, and some system board registers. Access to system memory or channel attached memory is determined by the translation control word mechanism. See "Translation Modes" on page 5-17 for details of the translation mechanism. No address translation occurs for alternate controller operations to devices in the I/O channel attached memory space or in the I/O channel I/O address space. Some system board registers can be protected from access from alternate controllers. See "DMA Access Authorization" on page 5-30 for information

on system board register access protection mechanism. The channel status register and translation control word matrix cannot be accessed by alternate controllers.

Alternate controllers connected to DMA channels 0-3 and 5-7 requires the channel which the alternate controller is connected to be programmed in 'cascade' mode. See "DMA Types" on page 6-16 for details of alternate controller operation.

DMA channel 8 must be enabled by writing to the channel 8 enable register when using an alternate controller on DMA channel 8.

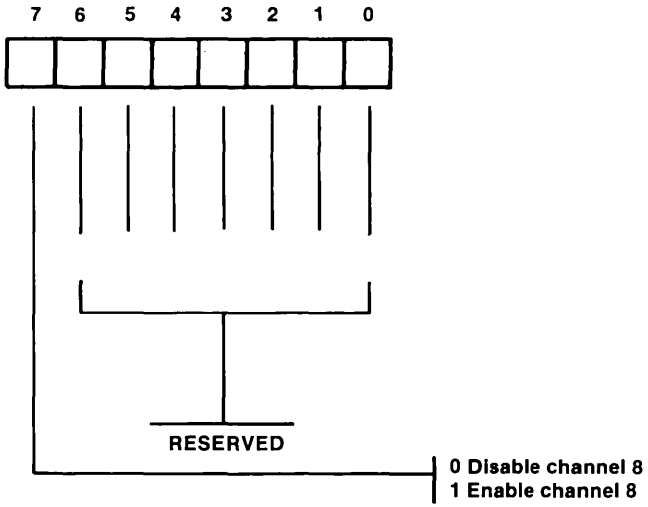


Figure 5-41. Channel 8 Enable Register(Address X'008C00')

Refresh

The system board provides refresh for I/O channel attached dynamic memory devices. Refresh can be enabled or disabled by writing to the channel control register. The refresh rate of one memory cycle every 6.4 or 14.9 microseconds can be selected by writing to the channel control register. Up to 16 refresh times can be buffered by the system board. This provides a maximum of 255 msec. latency. After 5 refreshes are buffered, refresh request is posted to the system arbiter.

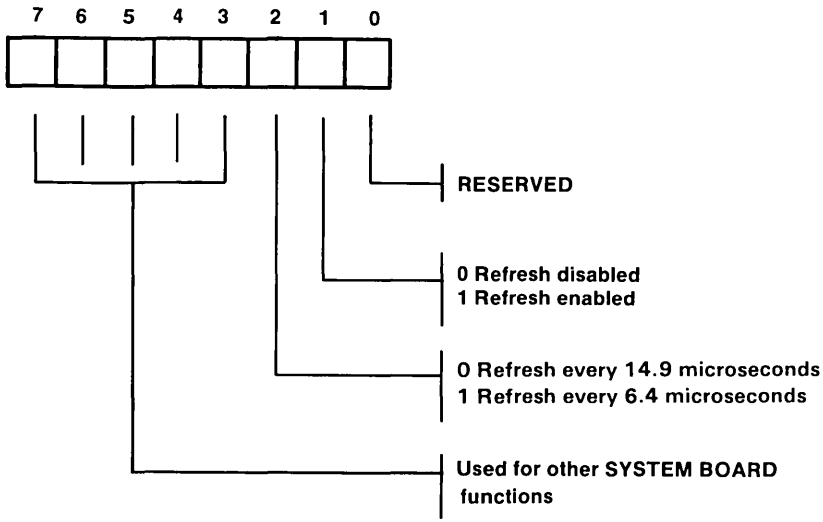


Figure 5-42. Channel Control Register (Address X'008C20')

Buffer Register

The buffer register can be read by or written to by the system processor or alternate controllers in the region mode. This register defines the bits enabled for each DMA channel. Only alternate controllers can do buffered mode transfers. Attempts by DMA devices to do buffered DMA transfers results in DMA exception errors.The Coprocessor may not use buffered mode.

When power is turned on, all bits in the buffer register must be set to the zero state.

Note:

Any bit = 1 Buffer mode enabled
Any bit = 0 Buffer mode disabled.

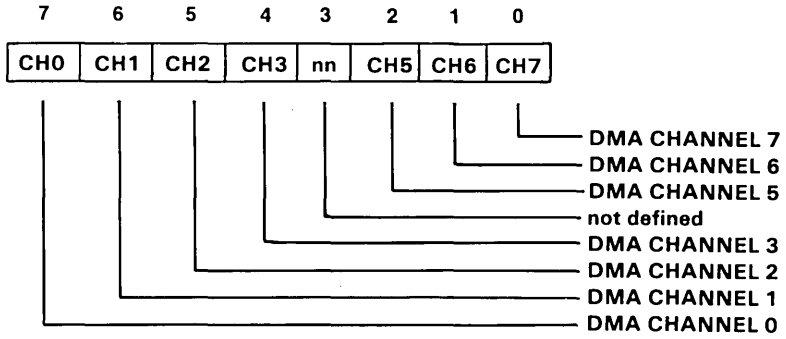


Figure 5-43. Buffer Register (Address X'0088C0')

Interrupt Controllers

The system board uses two 8259A interrupt controllers. These interrupt controllers support 11 interrupt request lines on the I/O channel and an additional 4 from sources on the system board. The 15 interrupt signals are divided into two groups. Each group of interrupt signals are processed by an 8259A controller which is attached to system processor interrupt level 3 or 4.

The 8259A interrupt controllers are run in the following mode:

- Single (not cascaded)
- Edge triggered
- Polled.

The 8259A accepts two types of command words from the processor. They are:

- Initialization Command Words (ICW1, ICW2, ICW3, ICW4)
- Operation Command Words (OCW1, OCW2, OCW3)

These are used for the initialization and operation of 8259A. See Figure 5-4 on page 5-12 for command word addresses .

The 8259A has three internal registers that are directly associated with the interrupt request signals.

Interrupt Request Register (IRR) is an 8-bit register which contains the levels that are requesting service. The highest level is reset from the IRR when an interrupt is acknowledged (polled) and being serviced.

The interrupt mask register (IMR) stores the mask bits for the interrupt lines to be masked in the IRR. A read OCW1 command will read the state of the IMR.

The in-service register (ISR) is an 8-bit register which contains the priority levels that are being serviced. A priority level is added to the ISR when a POLL command is issued and a priority level is removed or when End Of Interrupt (EOI) command is issued. The nonspecific EOI command resets the highest priority level while the specific EOI command resets a specified priority level.

Interrupt Priority

8259 Level	I/O Channel Interrupt Lvl	I/O Adapter	System Processor Interrupt Level
0	0 (Sys. Brd.)	8237 Terminal Count	3
1	10	I/O Channel IRQ 10	3
2	9	I/O Channel IRQ 9	3
3	3	I/O Channel IRQ 3	3
4	4	I/O Channel IRQ 4	3
5	1 (Sys. Brd.)	Keyboard Controller	3
6	2 (Sys. Brd.)	8530 Serial Port	3
7	7	I/O Channel IRQ 7	3

Figure 5-44. 8259 #1

8259 Level	I/O Channel Interrupt Lvl	I/O Adapter	System Processor Interrupt Level
0	8 (Sys. Brd.)	Reserved	4
1	11	I/O Channel IRQ 11	4
2	14	I/O Channel IRQ 14	4
3	12	I/O Channel IRQ 12	4
4	6	I/O Channel IRQ 6	4
5	5	I/O Channel IRQ 5	4
6	15	I/O Channel IRQ 15	4
7	13 (Sys. Brd.)	Serial Port Ex Ct1 IRPT	4

Figure 5-45. 8259 #2

Diagnostic Interrupt Activate Register

A Diagnostic Interrupt Activate (DIA) write command to the DIA register sets all 16 positions of the interrupt register to the activate (interrupting) state and holds them. A later write with data bit 0 reset (0) releases the interrupt requests. DIA tests all the interrupt levels.

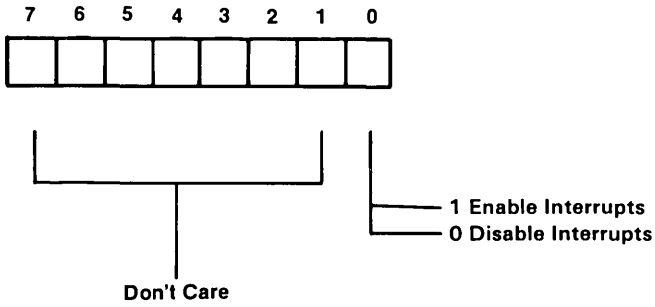
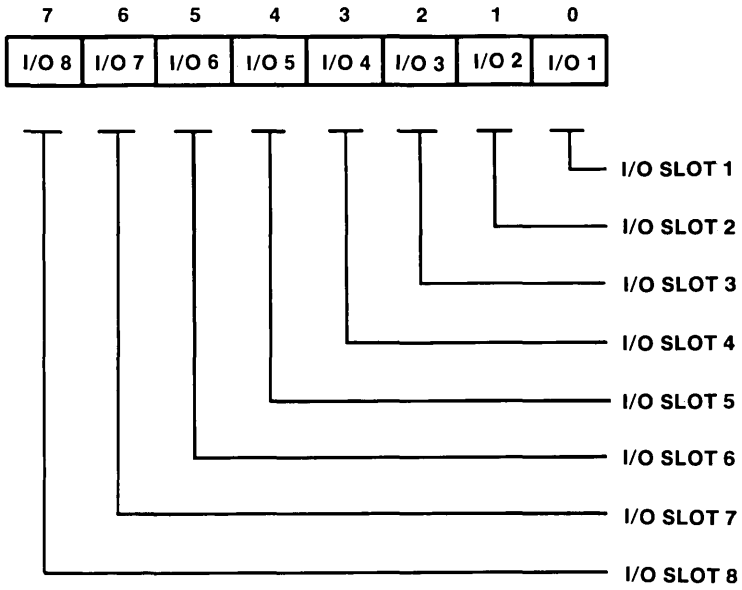


Figure 5-46. Diagnostic Interrupt Activate Register (Address X'008CA0')

Component Reset Register A (CRRA)

The outputs of this 8-bit register drives the resets to the system board I/O channel slots. The net effect of this register is to give each I/O channel slot the capability to be individually reset by software. This register can be read or written to by the processor board or an alternate controller operating in region mode. '-Power On Reset' going active causes the CRRA to activate all resets to the I/O channel slots.



Bit	Bit Definition	Comment
7	I/O Slot 8	(IBM 6150 only)
6	I/O Slot 7	(IBM 6150 only)
5	I/O Slot 6	All units
4	I/O Slot 5	All units
3	I/O Slot 4	All units
2	I/O Slot 3	All units
1	I/O Slot 2	All units
0	I/O Slot 1	All units

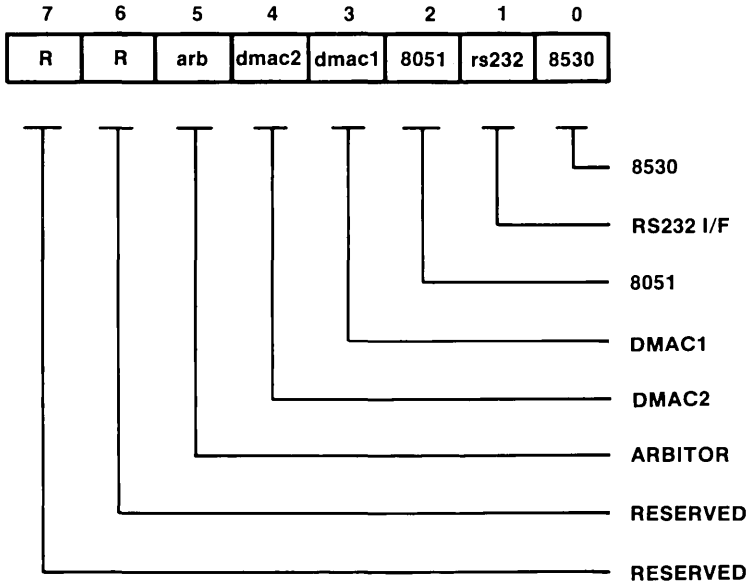
Figure 5-47. Component Reset Register A (Address X'008C40')

Note:

Any bit = 1 is enabled.
Any bit = 0 is disabled.

Component Reset Register B (CRRB)

The outputs of this 8-bit register drives the reset to the serial ports, keyboard adapter, 8237's and the system arbiter. This register can be read or written by the processor board or any alternate controller programmed in region mode. This register activates all resets during POR and a processor initiated reset.



Bit	Bit Definition
7	Reserved
6	Reserved
5	Arbiter
4	DMAC2 (Channel 5 - 7)
3	DMAC1 (Channel 0 - 3)
2	8051
1	RS232C I/F
0	8530

Figure 5-48. Component Reset Register B (Address X'008C60')

Note:

Any bit = 1 is enabled.
Any bit = 0 is disabled.

Channel Status Register (CSR)

This 32-bit register captures the status of the system board error conditions. This register can be read or written only by the processor board. Writes to the CSR resets all valid status bits.

Note: Reserved CSR bits are always 1 on CSR reads. The CSR register resides on the processor channel and is cleared by POR.

Processor Reg Data Image	CSR Bit	Bit Definition	CSR Bit Acronym	Initial State At Power On
MSB 0	31	Exception Reported	EXR	0
1	30	Interrupt Pending	INTP	0
2	29	Reserved	R	1
3	28	EPOW	EPOW	0
4	27	Soft Reset	SRST	0
5	26	System Attention	SAT	0
6	25	Reserved	R	1
7	24	PIO Error	PER	0
8	23	DMA Error-Channel 0	DE0	0
9	22	DMA Error-Channel 1	DE1	0
10	21	DMA Error-Channel 2	DE2	0
11	20	DMA Error-Channel 3	DE3	0
12	19	DMA Error-Channel 5	DE5	0
13	18	DMA Error-Channel 6	DE6	0
14	17	DMA Error-Channel 7	DE7	0
15	16	DMA Error-Channel 8	DE8	0
16	15	PIO/DMA	P/D	0

Figure 5-49 (Part 1 of 2). CSR - Channel Status Register Address (X'010800')

Processor Reg Data Image	CSR Bit	Bit Definition	CSR Bit Acronym	Initial State At Power On
17	14	Protection Violation	PVIO	0
18	13	Invalid Operation	INVOP	0
19	12	I/O Channel Check	IOCK	0
20	11	DMA Exception	DEXK	0
21	10	Channel Reset Captured	CRC	0
22	9	System Board Busy	SBB	0
23	8	PIO Request Pending	PRP	0
24	7	Reserved	R	1
25	6	Reserved	R	1
26	5	Reserved	R	1
27	4	Reserved	R	1
28	3	Reserved	R	1
29	2	Reserved	R	1
30	1	Reserved	R	1
LSB 31	0	Reserved	R	1

Figure 5-49 (Part 2 of 2). CSR - Channel Status Register Address (X'010800')

This section describes the error and interrupt conditions that occur within the IOCC. Each error or interrupt condition and the resulting response by the IOCC is described.

When an error is detected by the IOCC, a CSR bit is set to show which interface initiated the operation. A bit is also set indicating that the CSR contains valid error status, and that the processor interrupt is pending. If a second error is detected before software can clear the status word, the CSR will reflect the conditions of the first error and the error type.

The channel status register follows:

Bits 0-7 Always one.

Bit 8 PIO Request Pending. This bit is set when a reset or error occurs while a PIO to the IOCC is pending on the processor channel. If this bit is set when a processor timeout occurs, it indicates that the IOCC may have caused the error.

Bit 9 System Board Busy. This bit indicates that the IOCC was busy when a reset or error occurred. On reset it implicates the IOCC in a processor timeout.

Bit 10 Channel Reset Captured. This bit is set if the processor detects a hardware error while the IOCC had detected no errors. If this bit is set , then bits 16-31 reflect the IOCC state at the time the processor detects the error. When this bit is set, the state of bits 16 -31 is frozen as if an error had occurred.

Bit 11 DMA Exception. This bit is set if an exception reply is received by the IOCC during a DMA operation. Status on the active DMA channel is also latched when this error occurs.

Cause: Processor board returns an exception to a DMA adapter request to system memory.

Action: DMA Exception

1. Send level 2 processor interrupt
2. Latch CSR bits
   - DMA error for active channel = 1
   - PIO/DMA operation = 0
   - Interrupt pending = 1
3. Reset DMA controllers.

Bit 12 I/O Channel Check. This bit is set when the I/O channel check line is asserted by a device on the I/O channel. When this happens, the status of any PIO or DMA channel activity is latched.

Cause: A device on the I/O channel asserts the I/O channel check line.

Action:

1. PIO real/virtual read or write in progress
   - Send level 2 processor interrupt
   - Latch CSR bits
     - PIO error = 1
     - PIO/DMA error = 1
     - Interrupt pending = 1
     - I/O channel check = 1
2. DMA operation in progress
   - Send level 2 processor interrupt
   - Latch CSR bits
     - DMA error for active channel = 1
     - PIO/DMA Operation = 0
     - Interrupt pending = 1
     - I/O channel check = 1.

Note: The current I/O channel cycle will end normally.

Bit 13 Invalid Operation. This bit is set when an invalid PIO request occurs , or when a DMA adapter attempts an invalid operation (because of an invalid DMA setup). An example of this error is a DMA device attempting to use region mode. Status of the offending DMA channel is latched when this error occurs.

Cause:

1. Undefined PIO operation
2. A DMA device attempts to use region mode
3. A DMA device attempts to use buffered mode

Action:

1. Undefined PIO Operation
   - Send exception reply
   - Latch CSR bits:
     - PIO error = 1
     - PIO/DMA operation = 1
     - Invalid operation = 1
     - Exception reported = 1
2. DMA device error
   - Send level 2 processor interrupt
   - Latch CSR bits
     - DMA error for active channel = 1
     - PIO/DMA operation = 0
     - Interrupt pending = 1
     - Invalid operation = 1.

Note: Operation cancelled.

Bit 14 Protection Violation. This bit is set when a PIO command attempts to access a privileged level register from unprivileged state or when a alternate controller in region mode attempts access to a protected system board I/O address.

Cause:

1. PIO access to a privileged register from unprivileged state.
2. Attempt by an alternate controller operating in region mode to access I/O subsystem registers while CCR channel access bits prohibit it.

Action:

1. PIO read or virtual write induced error
   - Send exception reply
   - Latch CSR bits:
     - PIO error = 1
     - PIO/DMA operation = 1
     - Protection violation = 1
     - Exception reported = 1
2. PIO real write in progress
   - Send level 2 processor interrupt
   - Latch CSR bits
     - PIO error = 1
     - PIO/DMA operation = 1
     - Interrupt pending = 1
     - Protection violation = 1
3. Alternate controller protection violation
   - Send level 2 processor interrupt
   - Latch CSR bits
     - DMA error for active channel = 1
     - PIO/DMA operation = 0
     - Interrupt pending = 1
     - Protection violation = 1

Note: Operation cancelled.

Bit 15 PIO/DMA. This bit qualifies whether error status latched on the first occurrence of an error was because of a PIO or DMA operation. This bit is set when the first error was because of a PIO operation.

Bits 16-23 DMA Error 8-5, 3-0. These bits indicate the active DMA channel during an error. These bits are not locked after the first error.

Bit 24 PIO Error. This bit indicates an error during a PIO command. This bit is not locked after the first error.

Bit 25 Reserved. This bit is reserved and not guaranteed to be the same on successive CSR read commands.

Bit 26 System Attention Sequence from Keyboard. This bit is set when the system attention interrupt key sequence is detected by the keyboard controller. When active, this bit causes a processor level 0 interrupt.

Cause: System attention keystroke sequence detected by keyboard adapter.

Action:

1. Activate processor level 0 interrupt
2. System attention CSR bit = 1 (CSR is not locked).

Bit 27 Soft Reset. This bit is set when the soft reset key sequence is detected by the keyboard controller. When active, this bit causes a system reset. This bit provides the IPL software with the ability to differentiate between a power on reset and a soft reset.

Cause: Soft reset keystroke sequence detected by keyboard adapter.

Action:

1. Activate processor reset
2. Soft reset CSR bit = 1 (CSR is not locked until processor resets IOCC)
3. Processor resets IOCC
4. All current IOCC interface activity is cancelled
5. The CSR is frozen, logging current activity, if any.
   - Exception reported bit is reset
   - Interrupt pending bit is reset
   - Channel reset capture = 1 if no previous errors
   - PIO operation pending bit may be set
   - Executing PIO operation is cancelled if:
     - PIO error bit = 0
     - PIO/DMA bit = 1
     - System board busy bit = 1
   - Executing DMA operation is cancelled if:
     - DMA error bit = 1 for active channel
     - PIO/DMA operation bit = 0
     - System board busy bit = 1. (If DMA activity was on the processor channel at the time of the processor initiated reset).

Bit 28 Early Power Off Warning. This bit indicates a fault was detected in the AC power supply. When active, this bit causes a TRAP interrupt. A minimum of 2 ms. of good machine operation is guaranteed following detection of this signal.

Cause: Power supply detected loss of ac power

Action:

1. Activate trap signal
2. Set EPOW CSR bit (CSR is not locked).

Bit 29 Reserved. This bit is reserved and not guaranteed to be the same on successive CSR read commands.

Bit 30 Interrupt Pending. This bit indicates the IOCC has detected an error condition and generated an interrupt to the processor. This bit is not set for errors reported with exception replies.

Bit 31 Exception Reported. This bit indicates the IOCC has detected an error and reported it via the processor's exception mechanism. This bit is not set if the error is reported by interrupt.

Miscellaneous CSR Errors

Processor Initiated Reset

Cause:

1. Unrecoverable error detected on processor board.

Action:

1. Processor resets IOCC
2. All current IOCC interface activity is cancelled
3. The CSR is locked, logging current activity, if any.
   - Exception reported bit = 0
   - Interrupt pending bit = 0

- Channel reset captured = 1
- If PIO operation is cancelled;
  - PIO error bit = 0
  - PIO/DMA operation bit = 1
  - System board busy bit = 1
- If DMA operation is cancelled, DMA error for active channel = 1
  - PIO/DMA operation bit = 1
  - System board bit = 1
  - Software is responsible for resetting the I/O slot with the affected DMA adapter.

CSR Bit Status

These CSR bits are not affected by CSR locked status

- Interrupt pending
- Exception reported
- PIO error
- DMA error (Channels 0 - 8)
- EPOW
- Soft Reset
- System attention

These CSR bits are locked when an error occurs.

- PIO/DMA Operation
- Protection violation
- I/O channel check
- DMA exception
- Channel reset captured
- PIO request pending
- System board busy.

Memory Configuration Register

The 8-bit memory configuration register (MCR) provides the size and refresh rate information obtained from the configuration bits on the system memory boards. The MCR can be read by the system processor, or by an alternate controller in region mode. The MCR is at system address X'F0008C80'.

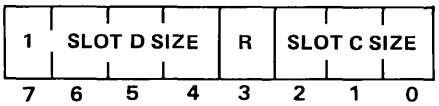


Figure 5-50. Memory Configuration Register

Bit
Definition

7 Reserved, always = 1.

6-4 Memory board size in slot D. These three bits specify the following memory board sizes.

000 4MB memory board
001 1MB memory board
010 Reserved
011 Reserved
100 8MB memory board
101 2MB memory board
110 512KB memory board
111 No memory board.

3 Refresh rate. If bit 7 is 0, system memory is refreshed every 7.0 usec. If bit 7 is 1, system memory is refreshed every 13.8 usec.

2-0 Memory board size in slot C. These three bits specify the memory board sizes shown above.

I/O Delay Register (IDR)

Writing to the I/O delay register result in the execution of a slow PIO cycle with data being lost. This register is provided for software to supply a mechanism to create reliable timing delays. This register can be written by the processor board or an alternate controller in region mode. The IDR is located at address X'0080E0'.

Writes to the I/O delay register results in delays of approximately 1 microsecond per byte transferred. This provides the following delays for each type of store instruction executed to the IDR by the system processor.

- Store character (1 byte transfer) - 1 μsec.
- Store halfword (2 byte transfer) - 2 μsec.
- Store word (4 byte transfer) - 4 μsec.

I/O Subsystem Adapters

The I/O subsystem adapters which reside on the system board are described in this section.

Real Time Clock

The RT PC system uses the MC146818 real time clock module to provide the real time clock function and provide 50 bytes of nonvolatile RAM. The real time clock runs on the battery when the system is turned off and runs on the power supply when the system is turned on. The oscillator input to the real time clock is 32.768 KHz. The clock out, square wave, and interrupt request outputs are programmable. The clock output is set to 32.768 KHz. The square wave output is programmed for 1.024 KHz and is used by the system processor interval timer. The interrupt request output is programmable and is connected to the system processor interrupt level 1. The nonvolatile RAM stores system configuration information. The I/O address space of this module is X '008800' through '00883F' and is shown in the following table.

I/O Address	Type	Comments
008800	R/W	Seconds
008801	R/W	Seconds Alarm
008802	R/W	Minutes
008803	R/W	Minutes Alarm
008804	R/W	Hours
008805	R/W	Hours Alarm
008806	R/W	Day of Week
008807	R/W	Date of Month
008808	R/W	Month
008809	R/W	Year
00880A	R/W	Register A
00880B	R/W	Register B
00880C	R/-	Register C
00880D	R/-	Register D
00880E-00883F	R/W	Non Volatile RAM (50 Bytes)

Figure 5-51. Real Time Clock Address Map

For more information on the real time clock read the Motorola specifications on the MC146818.

Serial Ports

Two asynchronous serial ports are contained on the IBM 6150 system boards; they are called channel A and channel B. The serial ports attach to the low order byte (7-0) of the I/O channel and are isolated from the external I/O channel by tri-state buffers. Both serial ports are contained within one Z8530 Serial Communications Controller (SCC). The Z8530 requires one control register for each serial port external to the Z8530 itself. These external registers contain control and status information about the serial interface. Each serial port can transmit and receive data in programmed I/O mode or transmit data in Direct Memory Access (DMA) mode. Each serial port also has an EIA RS232C-like interface. See "RS232C Interface" on page 5-83.

Z8530 Module

The functional description of the Z8530 module is not discussed in this document; only major points of interest about the Z8530 are given here. For more information about the Z8530 refer to:

1. *Zilog Data Book - 1982/83*
2. *Z8030/Z8530 SCC Serial Communication Controller Technical Manual - 1983.*

As stated earlier both serial ports are contained within the Z8530 module and each serial port is designed to run in asynchronous mode only.

Capabilities of the serial ports include:

- Two independent full-duplex channels
- Five-, 6-, 7-, or 8-bits per character
- One, 1.5 or 2 stop bits
- Odd, even or no parity
- Break detection and generation
- Parity, overrun, and framing error detection
- Transmit or receive up to 19,200 bits per second
- Baud rate generator in each channel.

The Z8530 is connected to the lower two address bits (0-1). Address bit 0 is connected to the A/B pin of the Z8530 and address bit 1 is connected to the D/C pin of the Z8530. The addresses associated with the two serial ports are given in Figure 5-52 on page 5-77.

The Z8530 has two TTL-compatible clock signals connected to it. The first clock signal is connected to the PCLK pin, and runs at a rate of 3.58 MHz. The second clock signal, which runs at a rate of 3.072 MHz, is connected to the RTxCA and RTxCB pins of the Z8530 module. The serial ports use the baud rate generator in each of the channels in the Z8530, so this 3.072 MHz clock drives the baud rate generator in each of the channels in the Z8530. Write register 11, which controls the different clocking schemes of the Z8530, should be set to a specific value for the serial ports to operate properly.

I/O Address	Facility
00 80 00	Z8530 Serial Port Channel B Control
00 80 02	Z8530 Serial Port Channel B Data
00 80 01	Z8530 Serial Port Channel A Control
00 80 03	Z8530 Serial Port Channel A Data
00 80 20	External Register A
00 80 40	External Register B
00 80 60	Z8530 Intack

Figure 5-52. Serial Port Address Map

This specific value is:

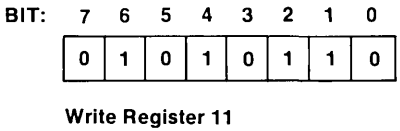


Figure 5-53. Write Register 11

Bit 7 is set to a logic level 0 since the RTxCA and RTxCB pins are connected to a TTL-compatible signal and not a quartz crystal. Bits 6 and 5 are set to 1 and 0 since the baud rate generator of the Z8530 drives the receive clock. Bits 4 and 3 are set to 1 and 0 since the baud rate generator of the 28530 drives the transmit clock. The TRxCA and TRxCB pins are not connected, so the values in bits 2, 1, and 0 are only recommended. Write register 11 is described in detail in Section 3.11 "Clocking Options" in the *Z8530 Technical Manual.* Another register to observe is write register 14. Bit 1 in write register 14 must be set to 0 since the baud rate generator is used to drive the transmit and receive clocks. How to use the baud rate generator is described in Section 3.9 "Baud Rate Generator" in the *Z8530 Technical Manual.*

Note: Since the baud rate generator is driven by a 3.072 MHz clock, the higher times clock modes can't be used for the higher transmit and receive data rates.

The Z8530 module is reset when both its RD and WR pins are driven to a logic level 0 at the same time. This reset is forced active by writing a 0 into bit-0 of the component reset register B. The reset is forced inactive by writing a 1 into bit-0 of the component reset register B.

The serial ports generate two interrupts, one by the external registers, which will be discussed later, and one by the Z8530. The interrupt generated by the Z8530 occurs on any condition that normally causes the Z8530 to generate an interrupt. This is described in Section 3.4 "Interrupts" in the *Z8530 Technical Manual.* The interrupt coming from the Z8530 is on level 6 of the first 8259. A special note about the Z8530 interrupt is the INTACK pin on the Z8530. One of the steps in clearing an interrupt within the Z8530 is to drive the INTACK pin to a logic level 0 and then drive the RD pin to a logic level 0. This step is done by driving a certain address onto the address bus and doing a programmed I/O read to the Z8530. This address is given in Figure 5-52 on page 5-77. Other pins of interest to the interrupt function of the Z8530 are IEI and IEO. The IEI pin is always held at a logic level 1, and IEO is not connected.

The serial ports can also transmit data in DMA mode, so the Z8530 must be put into DMA mode. The W/REQA pin generates the DMA request for channel A and the W/REQB pin generates the DMA request for channel B. Write register 1 must be set up a certain way for the Z8530 to perform in DMA mode. This is described in Section 3.5 "Block Transfer" in the *Z8530 Technical Manual.*

The last item observed on the Z8530 is that pins DTR/REQA, DTR/REQB, SYNCA, and SYNCB are all not connected.

Programmed I/O and DMA Modes

As mentioned earlier, the serial ports can transmit and receive data in programmed I/O mode or transmit data in DMA mode. Each serial port channel has two DMA channels as a DMA device assigned to it. Channel A of the serial ports can transmit data in DMA mode, using either DMA channel 0 or DMA channel 2. Channel B of the serial ports can transmit data in DMA mode using either DMA channel 1 or DMA channel 3. The external registers determine which serial port channel uses which one of the DMA channels; this is described later. Not only must the serial ports be initialized to perform DMA operations, but the translation control mechanism and the DMA control mechanism must also be initialized. When the serial ports are used in DMA mode, they operate on an 8-bit DMA channel.

External Registers

Each serial port has one, 8-bit external register that is not contained within the Z8530 module. These two external registers are identical except one is used for channel A and the other one is used for channel B. The addresses for external register A and external register B are shown in Figure 5-52 on page 5-77. The external registers have three main functions. They control part of the RS232C-like interface, they generate an interrupt for the serial port, and they enable the correct DMA channel for each serial port.

The external register maintains three of the RS232C-like interface signals. The external register determines whether Data Terminal Ready (DTR) is active or inactive, monitors DTR, monitors Data Set Ready (DSR), and monitors Ring Indicate (RI).

One interrupt to the interrupt controller can be generated by either or both of the external registers; this interrupt is on level 7 of the second 8259. Essentially, external registers A and B can generate an interrupt. These two interrupts are 'OR'ed together to form one interrupt signal going to the interrupt controller. An interrupt is generated when one of the following conditions occur on the RS232C-like interface:

Channel A DSR changes state from 0 to 1 or from 1 to 0

Channel A RI becomes active, changes from 0 to 1

Channel B DSR changes state from 0 to 1 or from 1 to 0

Channel B RI becomes active, changes from 0 to 1.

An interrupt is also generated for the following conditions if the serial ports are using DMA:

Channel A Terminal count is reached on the DMA channel being used

Channel B Terminal count is reached on the DMA channel being used.

Figure 5-54 on page 5-82 and Figure 5-55 on page 5-82 shows how the external registers A and B are arranged. The bit definition of each register follows:

Read Function Channel A

Bit 0 If this bit is a 1, DTR for channel A is active. If this bit is a 0, DTR is inactive.

Bit 1 If this bit is a 1, DSR for channel A is active. If this bit is a 0, DSR is inactive.

Bit 2 If this bit is a 1, RI for channel A is active. If this bit is a 0, RI is inactive.

Bit 3 If this bit is a 0, DSR has changed state or RI has become active on channel A. If this bit is a 0 and bit-5 of write function channel A is a 1, an interrupt is generated. If this bit is a 1, DSR or RI have not changed value.

Bit 4 If this bit is a 0, terminal count has been reached during a DMA operation to channel A of the serial ports. If this bit is a 0 and bit-5 of write function channel A is a 1, an interrupt is generated. If this bit is a 1, terminal count has not occurred.

Bit 5 Unused

Bit 6 Unused

Bit 7 Unused

Write Function Channel A

Bit 0 Setting this bit to a 1 activates DTR for channel A. Setting this bit to a 0 turns off DTR.

Bit 1 Setting this bit to a 1, allows serial port channel A to use DMA channel 0 for transmitting data. Setting this bit to a 0 disables DMA channel 0.

Bit 2 Setting this bit to a 1 enables serial port channel A to use DMA channel 2 for transmitting data. Setting this bit to a 0 disables DMA channel 2.

Bit 3 A programmed I/O write of a 1 to this bit clears the interrupt generated by DSR or RI. A programmed I/O write of a 0 to this bit causes no change.

Bit 4 A programmed I/O write of a 1 to this bit clears the interrupt generated when terminal count is reached during a DMA operation. A programmed I/O write of a 0 to this bit causes no change.

Bit 5 Setting this bit to a 1 allows an interrupt caused by DSR, RI, or terminal count to be driven to the interrupt controller. Setting this bit to a 0 disables an interrupt from being driven to the interrupt controller.

Bit 6 Unused

Bit 7 Unused

Read Function Channel B

Bit 0 If this bit is a 1, DTR for channel B is active. If this bit is a 0, DTR is inactive.

Bit 1 If this bit is a 1, DSR for channel B is active. If this bit is a 0, DSR is inactive.

Bit 2 If this bit is a 1, RI for channel B is active. If this bit is a 0, RI is inactive.

Bit 3 If this bit is a 0, DSR has changed state or RI has become active on channel B. If this bit is a 0 and bit-5 of write function channel B is a 1, an interrupt is generated. If this bit is a 1, DSR or RI have not changed value.

Bit 4 If this bit is a 0, terminal count has been reached during a DMA operation to channel B of the serial ports. If this bit is a 0 and bit-5 of write function channel B is a 1, an interrupt is generated. If this bit is a 1, terminal has not occurred.

Bit 5 Unused

Bit 6 Unused

Bit 7 Unused

Write Function Channel B

Bit 0 Setting this bit to a 1 activates DTR for channel B. Setting this bit to a 0 turns off DTR.

Bit 1 Setting this bit to a 1, allows serial port channel B to use DMA channel 1 for transmitting data. Setting this bit to a 0 disables DMA channel 1.

Bit 2 Setting this bit to a 1 allows serial port channel B to use DMA channel 3 for transmitting data. Setting this bit to a 0 disables DMA channel 3.

Bit 3 A programmed I/O write of a 1 to this bit clears the interrupt generated by DSR or RI. A programmed I/O write of a 0 to this bit causes no change.

Bit 4 A programmed I/O write of a 1 to this bit clears the interrupt generated when terminal count is reached during a DMA operation. A programmed I/O write of a 0 to this bit causes no change.

Bit 5 Setting this bit to a 1 allows an interrupt caused by DSR, RI, or terminal count to be driven to the interrupt controller. Setting this bit to a 0 disables an interrupt from being driven to the interrupt controller.

Bit 6 Unused

Bit 7 Unused

The external registers can be reset by activating bit-1 of the component reset register B (setting bit 1 to a 0). When this bit is active:

- DTRA and DTRB become inactive
- All the DMA channels to the serial ports are disabled
- All the causes of the external register interrupts are cleared
- The enable for generating an interrupt for both external registers is disabled.

In terms of bits for both registers:

Read Function

```
Bit 0 is set to 0.
Bits 3 and 4 are set to 1.
```

Write Function

```
Bits 0, 1, and 2 are set to 0.
Bit 5 is set to 0.
```

Deactivating bit 1 (setting bit 1 to a 1) of the component reset register B releases the reset to the external registers.

DMA Initialization Sequence

When the serial ports are being initialized to operate in DMA mode, a special initialization sequence must be followed to insure that the DMA works properly. There are four steps and they must be done in this order:

1. Completely initialize the serial ports, the translation control mechanism, and the DMA control mechanism, but keep the DMA channel being used disabled in the external register of the serial ports, keep the DMA channel being used masked in the mask register of the 8237, and leave bit 7 (WAIT/DMA request enable) of write register 1 in the Z8530 disabled.
2. Enable the DMA channel being used in the external register of the serial ports.
3. Unmask the DMA channel being used in the mask register of the 8237.
4. Enable bit 7 of write register 1 in the Z8530.

The sequence described above must be carried out to allow the DMA control mechanism to see the DMA request transition from an inactive state to an active state.

Bit	Read Function	Write Function
0	DTRA	DTRA
1	DSRA	EN DMA 0
2	RIA	EN DMA 2
3	- EXT IRQ	EXT IRQ Reset
4	- TC IRQ	Reset TC IRQ
5	Reserved	EN IRQ
6	Reserved	Reserved
7	Reserved	Reserved

Figure 5-54. External Register A

Bit	Read Function	Write Function
0	DTRB	DTRB
1	DSRB	EN DMA 1
2	RIB	EN DMA 3
3	- EXT IRQ	EXT IRQ Reset
4	- TC IRQ	Reset TC IRQ
5	Reserved	EN IRQ
6	Reserved	Reserved
7	Reserved	Reserved

Figure 5-55. External Register B

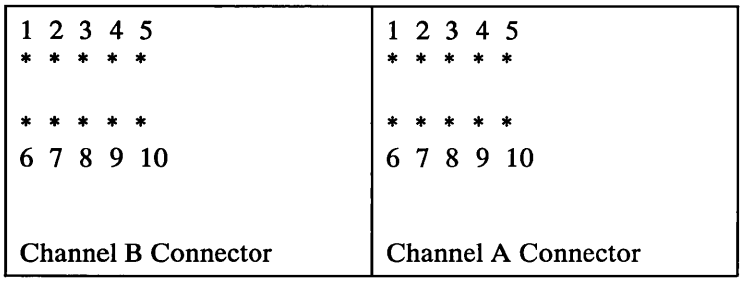
RS232C Interface

Each serial port supports the following RS232C interface signals:

- Transmit Data (TXD)
- Receive Data (RXD)
- Data Terminal Ready (DTR)
- Data Set Ready (DSR)
- Clear to Send (CTS)
- Request to Send (RTS)
- Data Carrier Detect (DCD)
- Ring Indicator (RI)
- Signal Ground

The serial ports use Texas Instrument SN75150 line drivers and SN75154 line receivers for its RS232C-like interface. Each serial port has a ten pin connector on the system board, and the pin configuration for channel A's connector is identical to channel B's connector. The serial port connector is shown in Figure 5-56 on page 5-84. The view given is as if someone were facing the two system board serial port connectors.

Note: From this view the channel B connector is to the left of the channel A connector.



Pin	Signal
1	TXD
2	DTR
3	RTS
4	RI
5	No Connection
6	RXD
7	DSR
8	CTS
9	DCD
10	Signal Ground

Figure 5-56. Serial Port Connectors

Note: The two connectors are identical.

Keyboard, Locator, Speaker Adapter

The keyboard, locator, and speaker adapter (referred to hereafter as the adapter) contains an 8051 single chip microcontroller programmed to support a serial keyboard interface, a locator device with a full duplex serial interface (UART), and a speaker. The adapter interfaces bidirectionally to the I/O channel through an 8-bit 8255A programmable peripheral interface chip, and a 6-bit command register.

The adapter supports the keylock switch which inhibits input from the keyboard and locator devices. The adapter also performs a multikeystroke detection, which directly initiates a system reset or a system attention interrupt, depending on the sequence detected.

The adapter resides on the system board. The system board provides the connectors to the keyboard, locator device, speaker, and keylock switch.

System Software Interface

System-to-adapter communications are over the two low-order data bus bytes (bits SD0-15) using I/O write operations. Adapter-to-system communications are initiated by the adapter raising an interrupt level request. Data transfers to the system are over the low-order data bus byte (bits SD0-7) using I/O read operations. Additionally, an adapter selective reset may be initiated by the system using an I/O write sequence.

8255 Programmable Peripheral Interface

The 8255A (referred to hereafter as 8255) is a general purpose programmable I/O chip used to interface the 8051 to the system and as a diagnostic sensing port. The 8255 is configured to operate in mode 2 with Ports B and C (lower) defined as inputs. Port A provides a bidirectional bus with the 8051, while Port C (upper) provides the necessary hand-shake controls.

8051 Microcontroller Functions

The adapter generally does not interpret information passing between the system and the keyboard or locator. That information is simply passed through the adapter. The 8051 receives data from the keyboard or locator, validates it, and passes it on to the system. The 8051 receives device commands from the system and passes them on to the selected device. The 8051 also receives commands from the system which are validated and executed by the 8051, such as reading and writing shared RAM.

The 8051 has 128 bytes of Random Access Memory (RAM). Commands are defined which allow the system to read or write a selected set of these bytes. This shared RAM contains status and mode control information, Reliability, Availability and Serviceability (RAS) activity and error logs, and device control parameters.

Keyboard Interface

The adapter receives a frame of data serial by bit from the keyboard, validates the frame, buffers up to 5 bytes, and presents the data to the system as a byte of data in the 8255 input buffer. The adapter interrupts the system when data is placed in the interface buffer.

The system sends keyboard commands to the keyboard by writing to the adapter's command register and 8255 output buffer. The 8051 imbeds the data byte in a frame, then sends the frame serial by bit to the keyboard with odd parity.

The 11-bit framing protocol and keyboard commands are defined in the keyboard section of this manual. Scan codes received from the keyboard are passed on to the system in their original form, the adapter performs no translation. Keystrokes received while the buffer is full (overrun) are lost.

Locator Interface

The adapter contains a duplex serial port operating in an 11-bit UART framing protocol that communicates with a locator device. The adapter buffers up to 2 reports of locator device data. The system sends commands through the adapter to the locator. The data reports and commands are defined in the locator section in this manual. Reports received from the locator are passed on to the system in their original form, the adapter does not change the report byte content. Reports received while the buffer is full (overrun) are lost.

Speaker Interface

The keyboard contains a small speaker, which can be driven from either the adapter or the Coprocessor. The adapter always has control of the volume. Frequency and duration parameters are controlled by either the adapter or the Coprocessor through an OR circuit. System software must ensure that only one path is active at a time.

The adapter provides an automatic click tone through the speaker for every keystroke at a default frequency and duration.

The adapter buffers one set of frequency and duration parameters while a current tone is active.

RAS and Security Functions Provided by the Adapter

The adapter is able to force system attention interrupt and system reset signals to the system. These signals are initiated by two unique three-keystroke sequences. The system attention interrupt default keystroke sequence may be redefined or disabled by system software. The system reset keystroke sequence cannot be altered, and is always active with two exceptions:

- When the keylock switch is on
- The system software has disabled the keyboard interface or keyboard scanning.

The adapter monitors a signal from an external keylock switch. When the switch is on, all input from the keyboard or locator is suppressed in the adapter, including the special keystroke sequences. System IPL Read Only Memory (ROM) code interrogates the state of the keylock switch during its power-on initialization and acts appropriately if found on.

Individual activity counters maintain counts of frames received and transmitted thru the keyboard and locator ports. Retry counters accumulate counts of keyboard retries performed. Individual error counters accumulate the number of hard errors for the keyboard and locator.

The keyboard and speaker ports may be diagnostically wrapped or sensed at the signal points which leave the system board. No external wrap connectors are required. Also, UART port signals may be diagnostically sensed. The 8051 performs a self-test function after a system reset. Self-tests may also be performed on command from the system.

The adapter initiates retry operations with the keyboard on transmit and receive errors. The number of retries performed may be redefined by system software. The adapter returns 8051-detected error conditions to the system where additional error recovery procedures may be performed.

Adapter I/O Operations

System software communicates with the adapter's 8255 using memory mapped reads and writes. Configuring the 8255 and enabling or disabling interrupt requests is done with a write operation to the 8255. The adapter hardware reset function is performed by writing to the system's component reset register B.

Communications between the system software and the adapter's 8051 can be initiated by either party. The system can send 2 bytes of command and data to the adapter. The adapter has two modes of transferring data to the system, non blocked (single byte) and blocked (multiple bytes in sequence). The adapter signals the system when it has information to transfer with an interrupt request, if enabled. Alternatively, the system disables the interrupt request and polls the 8255 for data.

The following paragraphs list the specific, detailed steps used for communications between the system software and the adapter's 8051.

System Initiated Transfer to Adapter

1. System verifies that 8255 output buffer is not full (by testing its internal flag)
2. System sets an internal programming flag indicating 8255 output buffer full (OBF); (system should reset flag in an acknowledgement interrupt handler)
3. System issues I/O Write to adapter with command and data bytes
4. 8255 sets OBF [PC7=0], which initiates an 8051 interrupt
5. 8051 interrupt reads the command register (command must be read before data)
6. 8051 interrupt reads the data from 8255 PA reg
7. 8255 resets OBF [PC7=1], dropping interrupt to 8051
8. 8051 interrupt posts command and data to a work queue
9. 8051 exits interrupt level
10. 8051 command execution work queue initiates an interrupt request to system signifying execution status of the command:
    - ID 0 = Informational (cmnd accepted or rejected)
    - ID 3 = Returning requested byte
    - ID 4 = Requested block transfer ready

Adapter Initiated Transfer to System

1. 8051 waits for IBF=0 (input buffer empty) by testing PC5
2. 8051 writes interrupt ID to 8255 PC bits 2-0 (PC must be written before PA)
3. 8051 writes data byte to 8255 PA reg (input buffer)
4. 8255 sets IBF (input buffer full) which raises IRQ to system
5. System accepts interrupt request
6. System reads interrupt ID from 8255 PC reg (PC must be read before PA); if ID=B'100', then go to block transfer chart
7. System reads data byte from 8255 PA reg (Input Buffer)
8. 8255 resets IBF (input buffer empty) which drops IRQ
9. System processes information and exits interrupt level

Adapter Initiated Block Transfer to System

This sequence transfers multiple bytes of the following types of information:

- Locator data when in blocking mode
- Adapter shared RAM dump
- RAS Logs dump

1. 8051 waits for IBF=0
2. 8051 writes interrupt ID '100' to PC bits 2-0
3. 8051 writes block length (byte count) to 8255 PA Input Buffer
4. 8255 sets IBF=1 which raises IRQ to system
5. 8051 waits for IBF=0
6. System accepts interrupt request
7. System reads interrupt ID B'100' from 8255 PC reg
8. System reads count from 8255 PA reg
9. 8255 resets IBF which drops IRQ
10. System waits (polls) for IRQ to be set (either by sensing IRQ internally in the system processor, or by reading 8255 PC bit 5)
11. 8051 writes interrupt ID appropriate to data to 8255 PC bits 2-0 (B'010' or '011')

12. 8051 writes data byte to 8255 PA Input Buffer
13. 8255 sets IBF=1 which sets PC5 and raises IRQ to System (if enabled).
14. 8051 waits for IBF=0
15. System reads interrupt ID from 8255 PC reg
16. System reads data byte from 8255 PA reg
17. 8255 resets IBF which resets PC5 which drops IRQ
18. Repeat previous 8 steps for the number of data bytes indicated by count
19. System processes information and exits interrupt level.

IOW and IOR Operations

The following chart lists the defined I/O operations to the adapter:

Host OP'N	I/O Address	Function	Comments
IOR	8404	Read 8255 PA	Input buffer returned
IOR	8405	Read 8255 PB	Diagnostic sense
IOR	8406	Read 8255 PC	Low 3 bits = interrupt ID
IOW	8400	Write 8255 PA	Command and data latched for execution by 8051
IOW	8407	Configure 8255	Required data = X'C3'
IOW	8407	Enable IRQ	Required data = X'09'
IOW	8407	Disable IRQ	Required data = X'08'

Host OP'N	I/O Address	Function	Comments
IOW	8C60	Activate adapter reset	Required data = B'xxxx x0xx'
IOW	8C60	Release adapter reset	Required data = B'xxxx x1xx'

Read 8255 PA Input Buffer

Op	Addr	High-Data Byte	Low-Data Byte
IOR	8404		PA Input Buffer

Action

The 8-bit 8255 PA input buffer contents are returned to system software. 8255 PC bit 3 is reset, which drops IRQ to the system processor, and frees the 8255 PA input buffer.

Pre-condition

The 8255 PA input buffer is valid only when the 8255 PC bit 3=1, which initiates an interrupt request to the system processor.

Comments

The meaning of the returned byte is determined by the interrupt ID in PC bits 2-0, which must have previously been read.

Read 8255 PB Port

Op	Addr	High-Data Byte	Low-Data Byte
IOR	8405		Port B Inputs

Action

The following 8 signals wired into the 8255 Port B are sensed and returned to system software:

- Bit 7 = +Cpspkr/speaker frequency
- Bit 6 = +Speaker volume bit 1

- Bit 5 = +Speaker volume bit 0
- Bits 4-1 = Diagnostic sense of non-adapter system board signals
- Bit 0 = UART RXD signal

Pre-condition

(None)

Comments

This operation is primarily intended for diagnostic purposes. It may be issued at any time to dynamically sense the state of the 8 signals listed.

Read 8255 PC Register

Op	Addr	High-Data Byte	Low-Data Byte
IOR	8406		PC Reg Contents

Action

The 8-bit 8255 PC register content is returned to system software. PC bit 7 indicates whether PA output buffer is full (0) or empty (1). PC bit 5 indicates whether PA input buffer is full (1) or empty (0). PC bits 6 and 4 are used for hand-shaking controls between the 8255 and the 8051 and can be ignored. PC bit 3 is the interrupt request line to the system processor. PC bits 2-0 define 8 possible interrupt identifier codes for the PA input buffer as follows:

PC 7-0 (binary)	**Interrupt ID**
xx1x i000	0
xx1x i001	1
xx1x i010	2
xx1x i011	3
xx1x i100	4
xx1x i101	5
xx1x i110	6
xx1x i111	7

Where x = don't care bits
i = 1 if IRQ enabled, or 0 if disabled

Refer to "Adapter Initiated Interrupt Request - ID Codes" on page 5-96 for an explanation of the interrupt ID's.

Pre-condition

PC Reg bits 2-0 are valid only if PC bit 5=1. The 8255 resets PC bit 5 when the PA input buffer is read.

Comments

This operation determines why interrupt request was initiated by the adapter and how the PA input buffer should be interpreted. This operation must be issued prior to reading the PA input buffer. This operation may also be used prior to writing to the PA output buffer to determine if it is full or empty.

Write 8255 PA Output Buffer

Op	Addr	High-Data Byte	Low-Data Byte
IOW	8400	8051 Command	8051 Data

Action

Two bytes of data are written to the adapter as follows (bit 15 is most significant bit):

- Bits 15-14 - Should be 00
- Bits 13-8 - Latched in adapter command register
- BITS 7-0 - Latched in 8255 PA output buffer

The 8255 resets PC bit 7 (to 0) indicating that the PA output buffer is full, which initiates an interrupt request to the 8051. When enabled, the 8051 interrupt handler reads the low 5 bits from the command reg (bits 12-8), then the 8 bits from the 8255 PA output buffer (bits 7-0). The 8255 sets PC bit 7, (to 1) indicating that the PA output buffer is now empty. The 13-bit command and data is posted to an 8051 internal queue for subsequent execution.

Pre-condition

The 8255 PA output buffer must be empty. This is determined by interrogating 8255 PC bit 7: 0 = full, 1 = empty.

Comments

The 13-bit command and data combinations are described in "Adapter Commands" on page 5-98. The next 8051 execution handler will acknowledge the command by initiating an appropriate Interrupt ID.

Configure 8255

Op	Addr	High-Data Byte	Low-Data Byte
IOW	8407		X'C3'

Action

The 8255 resets its internal registers (Control, PA, PB, PC). Then the data byte is loaded into the 8255 control register, setting the 8255 to operate as follows:

- PA is a bidirectional port with an output buffer and an input buffer.
- PB is an input-only sensing port for diagnostic use.
- PC bits 7-4 are handshaking controls for PA to the 8051.
- PC bit 5 = 8255 input buffer full
- PC bit 3 is an interrupt request to the system processor.
- PC bits 2-0 are input-only sensing pins set by the 8051.

Pre-condition

The 8051 is held in its reset state while this configure is being done.

Comments

This operation is normally issued after the adapter has been reset, either by a reset adapter operation, system reset, or power-on.

Enable IRQ

Op	Addr	High-Data Byte	Low-Data Byte
IOW	8407		X'09'

Action

The 8255 PC bit 3 initiates an interrupt request to the system processor when the PA input buffer has been loaded by the 8051.

Pre-condition

The 8255 must be properly configured.

Comments

This operation is normally issued after the 8255 has been configured, and after interrupt request had previously been disabled.

Disable IRQ

Op	Addr	High-Data Byte	Low-Data Byte
IOW	8407		X'08'

Action

The 8255 PC bit 3 cannot initiate an interrupt request to the system processor.

Pre-condition

(None)

Comments

This operation suspends system software and adapter communications. Adapter and device operations are not directly affected. Overrun conditions may occur if the suspension lasts longer than the buffering capability of the devices attached, such as the keyboard and locator.

Adapter Reset Operation

Resetting the adapter is a two-step process, the reset must be activated, then released. The minimum time that the reset must be active is 10 microseconds. There is no maximum time.

Activate Adapter Reset

Op	Addr	High-Data Byte	Low-Data Byte
IOW	8C60		B'xxxx x0xx'

Action

The selective reset to the adapter is raised and held active. The 8051 is held reset. (The 8255 is only reset by power-on.)

Pre-condition

(None)

Comments

The adapter is held reset until a following release adapter reset operation is performed.

Release Adapter Reset

Op	Addr	High-Data Byte	Low-Data Byte
IOW	8C60		B'xxxx x1xx'

Action

The selective reset to the adapter is dropped. The 8051 performs its internal self-testing and initialization. At the completion of the 8051 self-testing, a completion code is posted to shared RAM address X'1C' and to the system with an interrupt ID 6.

Pre-condition

The adapter's selective reset was started either by an activate adapter reset operation, or by a system reset, or by a power-on. The 8255 must be configured before reset is released.

Comments

The 8051 performs its self-test and initializes RAM to the defined defaults. Refer to "Adapter and Keyboard Initialization Procedure" on page 5-141 for the recommended procedure to initialize the adapter and keyboard.

Adapter Initiated Interrupt Request - ID Codes

When the adapter (8051) has information to pass to the system, a 3-bit identification code is placed in the 8255 PC register bits 2-0 and a data byte in the PA input register. Then an interrupt request is raised to the system, if enabled. The following chart lists the interrupt ID codes, and their meanings:

Interrupt ID Number	Interrupt Meaning	Data Byte Contents
0	Informational interrupt	Information code*
1	Byte received from keyboard	Received byte
2	Byte received from UART device	Received byte
3	Returning byte requested by system	Requested byte
4	Block transfer ready	Byte count
5	Unassigned	
6	8051 self-test performed	Completion code
7	8051 detected an error condition	Error code**

* Refer to "Adapter Informational Codes Returned to System" on page 5-126.

** Refer to "Adapter Error Codes Returned to System" on page 5-127.

Adapter Commands

Commands interpreted by the 8051 are initiated by the system via an I/O write to the 8255 PA register (output buffer). Command byte bits 4-0 are latched in the adapter's command register and are decoded by the 8051 to determine the meaning of the data byte latched in the 8255 PA register. Command byte bits 7-5 are diagnostic controls with bit 5 latched in an extension to the command register. The diagnostic control bits must normally be B'000'.

Command Byte Decodes

The following chart lists the command byte decodes:

Cmnd Byte 7654 3210	Command Function	Data Byte Function
0000 0000	Select extended command set	Extended command (see next chart)
0000 0001	Write to keyboard	Byte to be transmitted to keyboard
0000 0010	Write to speaker	Tone duration low-byte (number of 1/128 sec ticks)
0000 0011	Write UART - control (no response)	Byte transmitted to UART device
0000 0100	Write UART - query (response expected)	Byte transmitted to UART device
0000 0101	Set UART baud rate	Baud rate counter value (default is 9600 BPS)
0000 0110	Initialize UART framing	Odd/even parity control and blocking factor
0000 0111	Set speaker duration	Tone duration high-byte
0000 1000	Set speaker freq-hi	Frequency counter high-byte

Cmnd Byte 7654 3210	Command Function	Data Byte Function
0000 1001	Set speaker freq-low	Frequency counter low-byte
0000 1010	Unassigned	
0000 1011	Unassigned	
0000 1100	Diagnostic write keyboard port pins	Bit 6 to kbd data out pin, bit 7 to kbd clock out pin
0000 1101	Unassigned	
0000 1110	Unassigned	
0000 1111	Unassigned	
0001 RRRR	Write shared RAM	Byte is written to 8051 shared RAM (low order 4 cmnd bits select shared RAM addr)

Extended Command Decodes

The extended command set is decoded by the 8051 from the data byte latched in the 8255 PA Register when the command register = X'00'. The following chart lists the data byte decodes for the extended command set:

Data Byte	Extended Command Function
00-1F	Read shared RAM (low order 5 bits = shared RAM addr)
2M	Reset mode bit M (M = X'0-F')

Data Byte	Extended Command Function
3M	Set mode bit M (M = X'0-F')
40-43	Initialize speaker volume: X'40' = Off, X'41' = Low, X'42' = Medium, X'43' = High
44	Terminate speaker and reset duration
45-4F	Unassigned
50	Invalid
51-53	Set scan count for system attention keystroke sequence X'51' = 1 Keystroke X'52' = 2 Keystrokes X'53' = 3 Keystrokes (default)
54-5F	Invalid
60-6F	Diagnose functions:
60	Execute 8051 soft reset - force abnormal end code X'A0'
61	Force system reset
62	Force system attention interrupt
63-6F	Unassigned diagnostics

Data Byte	Extended Command Function
70	Diagnostic sense keyboard and UART port pins returned byte: Bit 0 = UART RXD pin Bit 2 = Keyboard clock in pin Bit 3 = Keylock switch pin Bit 5 = Keyboard data in pin
71-7F	Unassigned diagnostics
80	Dump adapter shared RAM addr X'00-0F'
81	Dump adapter shared RAM addr X'10-1F'
82	Dump RAS logs and reset activity and error counters
83	Dump RAS logs without counter reset
84-8F	Unassigned
90	Restore initial conditions
91-DF	Unassigned
E0-EF	Read 8051 release marker (low order 4 bits = byte offset into 16-byte release marker)
FX	NOP - adapter only returns an ACK interrupt to system (X = Don't care)

Select Extended Command Set (X'00')

The adapter interprets the data byte as an extended command which does not require an associated parameter. Refer to the specific extended command description.

Write to Keyboard (X'01')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code indicating acceptance or rejection of the command:

X'00'	Command accepted
X'41'	Command rejected - Adapter is busy with a previous transmission to the keyboard (S3=1)
X'42'	Command rejected - Keylock switch is on and M13=1
X'43'	Command rejected - Keyboard interface is disabled (M11=0)
X'44'	Command rejected - Associated data byte is invalid

If the command is accepted, then the associated data byte is transmitted to the keyboard.

The associated data byte transmitted to the keyboard must not be the keyboard resend command as defined by shared RAM address '03'.

Write to Speaker (X'02')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code indicating acceptance or rejection of the command:

X'01'	Command accepted - Speaker started
X'04'	Command accepted - Parameters queued
X'47'	Command rejected - invalid duration specified
X'48'	Command rejected - invalid frequency count value specified
X'4A'	Command rejected - Speaker queue full (S7=1)

If the speaker queue is full, the command is rejected with code X'4A', otherwise the associated data byte is written to SRAM address X'02' as the pending duration-low and S7 is set on to indicate that speaker parameters have been queued. (Assume that pending duration-high and pending frequency have been previously set in SRAM addresses X'01', X'15', and X'16', respectively.)

If the speaker facilities are busy (S0=1 or S1=1), the command is accepted with a parameters queued acknowledgement. When the facilities become available, these queued parameters are validated and activated.

If speaker facilities are currently available, S7 is set off. The pending frequency and duration parameters are validated and, if invalid, an appropriate command reject code is returned. Valid speaker parameters are activated and the command is accepted with a speaker started acknowledgement.

(Refer to "Adapter Speaker Control" on page 5-117.

Write to UART Device (X'03' and X'04')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code indicating acceptance or rejection of the command:

X'00' Command accepted

X'42' Command rejected - Keylock switch is on and M13=1

X'4B' Command rejected - UART interface is disabled (M12=0)

X'4C' Command rejected - Adapter is busy with a previous write to UART (S4=1)

If the command is accepted, the associated data byte is transmitted to the UART (locator) device.

The adapter has dual internal buffers for blocking reports received from a locator device. Each buffer has a 6-byte length to receive a device report up to 6 bytes long. One buffer can be transmitting to the system while the other is receiving a report from the device. If the device attempts to start a third report while the two buffers are busy, that third report (and all following ones) are discarded until a buffer is available.

Mode bit M5 specifies whether blocking of received reports is active or inactive. When inactive (M5=0), a byte received from the device is passed on to the system. When blocking is active (M5=1), report bytes received are buffered and blocked according to the blocking factor in SRAM address X'19'. Blocked report bytes are only transferred to the system when the complete report has been received by the adapter.

If the adapter detects a parity error on a report byte being received from the locator, the adapter discards that report.

Write UART - Control (X'03')

All UART device commands not having a defined response must be issued with the Write UART - Control command to the adapter. (See 5-131 for particular device information about this command.)

Write UART - QUERY (X'04')

All UART device commands having a defined response must be issued with the Write UART - Query command to the adapter. An error has occurred if a device command requiring a response is sent, and no response is received within 25 milliseconds. The adapter returns the device error code X'EA' to the system. (See 5-131 for particular device information about this command.)

Set UART Baud Rate (X'05')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code of X'00' indicating acceptance of the command. The associated data byte is written to SRAM address X'1B' and is a counter value used to control the UART transmit and receive baud rate. The valid rate is determined by the UART device plugged. The default rate is 9600 bits per second.

The counter value in SRAM address X'1B' is related to the baud rate according to the formula:

$$\text{Counter value} = 256 - \frac{\text{Osc}}{192 * \text{Baud}}$$

Where: Counter value = positive number rounded to an integer < 256
Osc = 8051 Oscillator frequency (Hz)
Baud = desired baud rate (bits per second)

For an assumed Osc value of 9.216 MHz, valid baud rates and corresponding counter values can be tabulated as follows:

Baud Rate	Counter (dec)
24,000	254
9,600	251
4,800	246
2,400	236
1,200	216
600	176
300	96

Initialize UART Framing (X'06')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code indicating acceptance or rejection of the command:

X'00' Command accepted

X'4E' Command rejected - invalid framing parameter

If the command is accepted, the associated data byte is written to SRAM address X'19'. It controls the parity generation and checking for UART transmission and reception, and specifies the number of bytes in the report received from the UART device. The framing parameter is defined as follows:

Bit 7: 1= odd parity (default); 0= even parity
Bits 6-3: Must be zeros
Bits 2-0: Blocking factor; valid values are 2, 3, 4 (default), 5, or 6

Note: The blocking factor must match the report length of the locator device when blocking is active (M5=1).

Set Speaker Duration - High Byte (X'07')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code indicating acceptance or rejection of the command:

X'00' Command accepted

X'4A' Command rejected - Speaker queue full (S7=1)

If the command is accepted, then the associated data byte is written to SRAM address X'01' as the pending speaker duration - high byte. No validation of the duration value is performed until the speaker parameters are activated. Refer to "Adapter Speaker Control" on page 5-117.

Set Speaker Frequency - High/Low Byte (X'08' and X'09')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code indicating acceptance or rejection of the command:

X'00' Command accepted

X'4A' Command rejected - Speaker queue full (S7=1)

If the command is accepted, the associated data byte is written to SRAM address X'15' for high byte command X'08', or SRAM address X'16' for low byte command X'09', as the pending speaker frequency. No validation of the frequency value is performed until the speaker parameters are activated.

(Refer to "Adapter Speaker Control" on page 5-117.)

Diagnostic Write Keyboard Port Pins (X'0C')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code indicating acceptance or rejection of the command:

X'00' Command accepted

X'51' Command rejected - illegal mode (M10=0)

This command is executed only in diagnostic mode. Data byte bit 7 is written to the keyboard clock line and data byte bit 6 is written to the keyboard data line. These values are kept on the indicated keyboard interface lines until the interface is cleared, or a later diagnostic write keyboard port pins alters them.

Write Shared RAM (X'1R')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code of X'00' indicating acceptance of the command. The associated data byte is written to the SRAM address specified by the low 4 bits of the command byte, which address the read/write shared RAM addresses X'00' through X'0F'.

Extended Command Descriptions

Read Shared RAM (X'00' - X'1F')

The adapter acknowledges the command by returning an interrupt ID 3 and an associated data byte containing the contents of the SRAM address specified by bits 4-0 of the extended command byte. The command permits system software to read any single byte of read/write or read-only shared RAM addresses X'00' through X'1F'.

Reset Mode Bit (X'20' - X'2F')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code indicating acceptance or rejection of the command:

X'00' Command accepted

X'51' Command rejected - illegal mode (M10=0)

If the command is accepted, the designated mode bit in shared RAM is cleared (set to 0). The mode bit is determined by the decimal value of the extended command byte bits 3-0.

Resetting of mode bit M11 may only be done while in diagnostic mode (M10=1), else the command is rejected.

Set Mode Bit (X'30' - X'3F')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code of X'00' indicating acceptance of the command. The designated mode bit in shared RAM is set to 1. The mode bit affected is determined by the decimal value of the extended command byte bits 3-0.

Initialize Speaker Volume (X'40' - X'43')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code of X'00' indicating acceptance of the command. The speaker volume controls, and the associated mode bits 8 and 9, are set according to the specific command:

X'40' Speaker volume = Off; M9,M8 = 00
X'41' Speaker volume = Low; M9,M8 = 01
X'42' Speaker volume = Medium; M9,M8 = 10
X'43' Speaker volume = High; M9,M8 = 11

The volume controls remain set until altered by a following initialize speaker volume command, set and reset of mode bits 8 or 9, or an adapter reset.

Volume commands are immediately executed, volume is not a queued parameter.

Terminate Speaker and Reset Duration (X'44')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code indicating thc current state of the speaker:

X'02' Speaker was already inactive (S1=0); no action taken

X'03' Speaker was active and has been terminated

Speaker termination clears status bits S0 and S1, clears the speaker duration ticks in SRAM addresses X'13' and X'14', and quiesces the adapter's speaker frequency input signal (allowing other features to control the speaker frequency).

Status bit S7 is set off, indicating no speaker command queued. Any values in SRAM addresses X'01', '02', '15', and '16' are not affected.

Set Scan Count for System Attention Keystroke Sequence (X'5S')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational Code indicating acceptance or rejection of the command:

X'00' Command accepted

X'50' Command rejected - invalid count specified

If the command is accepted, the indicated scan count S is set in SRAM address '17' and the system attention keystroke sequence state is reset. The extended command byte bits 3-0 are interpreted as a scan count for the system attention keystroke sequence. Valid values for S are: 1, 2, 3 (default).

Execute 8051 Soft Reset (X'60')

If the adapter is not in diagnostic mode (M10=0), the command is rejected by returning an interrupt ID 0 with an informational code of X'51', illegal mode.

If the adapter is in diagnostic mode (M10=1), then an abnormal end code condition is forced. An interrupt ID 7 with an abnormal end code of X'A0' is returned to the system, plus two additional bytes, each with an interrupt ID 7. The adapter will then re-start the 8051, perform the self-tests, and report the self-test completion code with an interrupt ID 6. At the conclusion of this command, the adapter is in normal operations with SRAM set to the defined defaults and the device interfaces cleared. (Refer to 5-127) for abnormal end codes.

Force System Reset (X'61')

If the adapter is not in diagnostic mode (M10=0), the command is rejected by returning an interrupt ID 0 with an informational code of X'51', illegal mode.

If the adapter is in diagnostic mode (M10=1), an immediate system reset is forced. This command performs the same function initiated by the system reset special keystroke sequence.

Note: System software issuing this command will be terminated!

Force System Attention Interrupt (X'62')

If the adapter is not in diagnostic mode (M10=0), the command is rejected by returning an interrupt ID 0 with an informational code of X'51', illegal mode.

If the adapter is in diagnostic mode (M10=1), an immediate system attention interrupt is forced. This command performs the same function initiated by the system attention special keystroke

sequence except that a scan code is not queued, nor placed in SRAM address X'1A'. This command ends by returning an interrupt ID 0 with an informational code of X'00'.

Diagnostic Sense Keyboard and UART Port Pins (X'70')

If the adapter is not in diagnostic mode (M10=0), the command is rejected by returning an interrupt ID 0 with an informational code of X'51', illegal mode.

If the adapter is in diagnostic mode (M10=1), then an interrupt ID 3 is returned with an associated data byte defined as follows:

Bit 0 = UART Receive Data (RXD) input signal
Bit 1 = 0
Bit 2 = Keyboard Clock In input signal
Bit 3 = Keylock Switch input signal
Bit 4 = 0
Bit 5 = Keyboard Data In input signal
Bit 6 = 0
Bit 7 = 0

Dump Adapter Shared RAM (X'80' and X'81')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code indicating acceptance or rejection of the command:

X'00' Command accepted

X'60' Command rejected - RAM queue is busy with prior dump command

If the command is accepted, then a 16-byte block of shared RAM is queued for transmission to the system. Command X'80' queues the read/write SRAM addresses X'00' - '0F'; command X'81' queues the read-only" SRAM addresses X'10' - '1F'. When the queued block is ready for transmission to the system, an interrupt ID 4 with an associated data byte containing the byte-count (16) is returned. Then each SRAM byte is returned, in sequence, with an interrupt ID 3.

Later dump adapter SRAM or dump RAS log commands are rejected until the currently queued block has been transmitted.

Dump RAS Logs...With/Without Reset (X'82' and X'83')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code indicating acceptance or rejection of the command:

X'00' Command accepted

X'60' Command rejected - RAM queue is busy with prior dump command

If the command is accepted, then a 12-byte block of RAS logs information (SRAM addresses X'20' - '2B') is queued for transmission to the system. When the queued block is ready for transmission to the system, an interrupt ID 4 with an associated data byte containing the byte-count (12) is returned. Then each SRAM byte is returned, in sequence, with an interrupt ID 3. After the block

has been transmitted, SRAM addresses X'20' - '2B' are zeroed if the operation was initiated with Command X'82'. Otherwise, SRAM is not affected.

Later dump adapter SRAM or dump RAS logs commands are rejected until the currently queued block has been transmitted.

Restore Initial Conditions (X'90')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code of X'00' indicating completion of the command. Shared RAM addresses X'00' thru X'1B' are initialized to the defined default conditions. The speaker command queue is cleared and any active speaker operation is terminated. All queued keyboard and locator input data is cleared and the respective interface controls reset to initial conditions.

This command does not affect RAS logs nor the attached keyboard and locator devices. The 8051 self-tests are not performed. This command will not clear certain types of information which may be, or have been, queued for transmission to the system (such as error code, status report informational code, requested RAM block).

Read 8051 Release Marker (X'E0' - X'EF')

The adapter acknowledges the command by returning an interrupt ID 3 and an associated data byte containing the contents of the selected release marker byte in 8051 ROM. The particular byte is specified by bits 3-0 of the extended command byte. Each command allows system software to read the corresponding single byte of the 16-byte release marker field defined as follows:

Bytes 0 - 2 VVV: Version number of 8051 code (3-character ASCII)

Bytes 3 - 7 YYDDD: Date of 8051 Version, Year and Julian Day (5-character ASCII)

Bytes 8 - 11 SSSS: Reserved field (binary zeroes)

Bytes 12 - 13 KK: Reserved field (binary zeroes)

Bytes 14 - 15: ZZ: Check sum on complete 8051 ROM (16-bit binary number)

NOP (X'F0' - X'FF')

The adapter acknowledges receipt of the command with an interrupt ID 0 and an informational code of X'00' indicating acceptance of the command. The low 4 bits of the extended command byte are ignored. This command performs no other function.

Adapter Shared RAM

SRAM Address	Read/Write Shared RAM Data	Number of Bytes	Initialized Value
00	Keyboard acknowledge byte	1	FA
01 - 02	Pending speaker duration ticks	2	0000
03	Keyboard resend command	1	FE
04	Keyboard break code	1	F0
05	Maximum retry count before a keyboard hard error is reported	1	08
06	Keyboard echo command	1	EE
07	Click duration	1	36
08	Click Suppress Scan Code 1	1	12
09	Click Suppress Scan Code 2	1	59
0A	Click Suppress Scan Code 3	1	39
0B - 0C	Click frequency - high/low	2	0896
0D	Keystroke initiate system attention - scan 1 Also click suppress scan code 4	1	11

SRAM Address	Read/Write Shared RAM Data	Number of Bytes	Initialized Value
0E	Keystroke initiate system attention - scan 2 Also click suppress scan code 5	1	19
0F	Keystroke initiate system attention - scan 3 (default = any scan match)	1	FF

Modes and Status Bits in Shared RAM

SRAM Address Bit	M/S Bit#	Adapter Mode and Status Bits	Initialized Bit Value
10.0	M0	Report receipt of keyboard acknowledgment byte with an informational interrupt	0=No
10.1	M1	Report completion of UART transmit with an informational interrupt	0=No
10.2	M2	Report keylock switch change with an informational interrupt	0=No
10.3	M3	Report completion of speaker tone with an informational interrupt	1=Yes
10.4	M4	Unassigned	0

SRAM Address Bit	M/S Bit#	Adapter Mode and Status Bits	Initialized Bit Value
10.5	M5	Blocking of received UART bytes is active	1=Yes
10.6	M6	System attention keystroke sequence search is active	1=Yes
10.7	M7	Suppress click for defined scan code set	0=No
11.0	M8	Speaker volume bit 0	0=MED
11.1	M9	Speaker volume bit 1	1=MED
11.2	M10	Diagnostic mode is in effect	0=No
11.3	M11	Keyboard interface is enabled with clear	1=Yes
11.4	M12	UART interface is enabled with clear	0=No
11.5	M13	Honor keylock switch	1=Yes
11.6	M14	Inhibit keystroke auto-click	0=No
11.7	M15	Ignore UART input	0=No
12.0	S0	Speaker frequency timer busy	0
12.1	S1	Timeout timer busy	0
12.2	S2	Keylock switch is set	(sw val)
12.3	S3	Keyboard transmit is busy	0

SRAM Address Bit	M/S Bit#	Adapter Mode and Status Bits	Initialized Bit Value
12.4	S4	UART transmit is busy	0
12.5	S5	Click busy	0
12.6	S6	Unassigned	0
12.7	S7	Speaker queue full	0

Notes:

1. Mode bits may be altered using the set/reset mode bit command. Status bits are read-only. Mode bits 0-3 enable the corresponding bits in the status report informational codes (refer to 5-127).
2. Setting mode bit 11 or 12 clears the keyboard or UART interface, respectively, even if the mode bit was already set. Mode bit 11 may be cleared only in diagnostic mode. The keyboard interface should not normally be disabled.
3. Mode bit 15 = 1 causes the adapter to discard all data received by the UART port.

Read Only Shared RAM

SRAM Address	Read Only Shared RAM Data	Number of Bytes	Initialized Value
13 - 14	Active speaker duration ticks remaining	2	0000
15 - 16	Pending speaker frequency - high/low	2	0000
17	Scan count for system attention keystroke sequence (maximum number of scan codes is 3)	1	03
18	Keystroke sequence state High 4 bits = system attention sequence state Low 4 bits = system reset sequence state	1	00
19	UART framing: MSB = odd/even parity control (1=odd) Low 3 bits = blocking factor (2 to 6 valid)	1	84
1A	System attention scan code - actual third byte received in a 3-byte sequence	1	00
1B	UART baud rate (counter reload value) defaults to 9600 bps. See note 3.	1	FB
1C	Actual 8051 self-test completion code (value indicated is for the good machine). See note 2.	1	AE
1D - 1E	8051 abnormal end information	2	0000
1F	Error code for most recent interrupt ID 7. See note 1.	1	00

Notes:

1. Refer to "Adapter Error Codes Returned to System" on page 5-127.
2. Refer to the "Adapter Self-Test After a Power-On, System, or Adapter Reset" on page 5-125.
3. Refer to the "Set UART Baud Rate" on 5-103.

RAS Logs in Shared RAM

SRAM Address	Read Only Shared RAM Data - RAS Logs	Number of Bytes
20 - 21	Number of keyboard frames received divided by 16 (not a keystroke count)	2
22 - 23	Number of keyboard receive retries performed (includes those resulting in hard errors)	2
24	Number of keyboard receive hard errors	1
25	Number of keyboard frames transmitted	1
26	Number of keyboard transmit retries performed (includes those resulting in hard errors)	1
27	Number of keyboard transmit hard errors	1
28 - 29	Number of UART frames (bytes) received divided by 16	2
2A	Number of UART frames transmitted	1
2B	Number of UART receive errors (number of frames received with bad parity)	1

Notes:

1. Logs are read-only via the dump RAS logs commands. All counters are cleared on completion of the dump RAS logs with reset command, or an 8051 reset.
2. Each counter is an 8-bit or 16-bit binary value. Because of the large number of keyboard or UART frames received, those two counters are incremented only once per 16 frames received.

Adapter Speaker Control

The speaker resides in the keyboard. However, the speaker is not controlled by the keyboard. It is completely controlled and powered by two signal wires through the keyboard cable to the system board. The speaker interface on the system board contains circuits to control the volume of the frequency signal. Both the adapter and the system's Coprocessor feature have access to that speaker interface, but only the adapter controls volume.

The adapter carries out the logical speaker functions used by system software, including frequency, duration, and volume parameters. The speaker functions are command-driven and queued. The adapter also carries out a keystroke click tone for the keyboard.

Sharing of Speaker Input With Coprocessor

The Coprocessor can activate the speaker signal through the interface circuits on the system board. This is a digital oscillator signal that is changing state at the rate for which the speaker is to be driven. This signal is 'ORed' with a similar signal from the adapter. The quiescent state of the signal is a high voltage level (logical 1). Either the adapter or Coprocessor can pull it low. No form of interlocking is provided to prevent one source from driving the speaker signal while the other source is driving it. The Coprocessor has no access to the volume control circuits in the speaker interface.

Speaker Frequency Control

The speaker frequency is set by system software. Commands send a 2-byte count value to the adapter. The adapter uses the count value to initialize internal counters, which control a speaker frequency signal driving the adapter's input to the speaker interface 'OR' circuit. System software sends the count value to the adapter using the set speaker frequency commands with high byte and low byte values sent separately.

The permissible range of the count value provides a speaker input frequency range from a maximum of 12,019 Hz to a minimum of 23 Hz.

The 2-byte count value used by the adapter is related to the generated speaker frequency according to the following formulas:

$$N = 11 - \log(9216000 * F / OSC)$$

$$Cx = 2 \exp [N]$$

$$Ct = \frac{OSC / (24 * F) - K0}{Cx} - K1$$

Where:

Cx	=	Count value, high byte
Ct	=	Count value, low byte
log	=	Base 2 logarithm function
F	=	Desired speaker frequency (Hz)
OSC	=	8051 oscillator frequency (Hz)
[N]	=	The exponent on the number 2 where the brackets mean that N is to be rounded to an integer and if [N] > 0, then use [N], else use 0
K0	=	3.7 (a constant)
K1	=	9.25 (another constant)

Limitations:

Cx	=	Element of the set: (1, 2, 4, 8, 16, 32, 64)
Ct	=	Integer in the range [19, 255] for Cx = 1 or, in the range [120, 255] otherwise

Exception:

A Cx value of 0 provides a silent tone. See the following note.

Example:

F	=	2500 Hz	=	Desired speaker frequency
OSC	=	9.216 MHz	=	Assumed 8051 oscillator
N	=	-0.29, so [N]	=	0
Cx	=	1		
Ct	=	141		

For the assumed OSC frequency of 9.216 MHz, the formulas for Cx and Ct can be simplified and tabulated as follows:

Frequency Range (Hz)	Cx (Dec)	Ct Formula
23 - 45	64	(6000 / F) - 9.31
46 - 90	32	(12000 / F) - 9.37
91 - 181	16	(24000 / F) - 9.48
182 - 362	8	(48000 / F) - 9.71
363 - 724	4	(96000 / F) - 10.18
725 - 1432	2	(192000 / F) - 11.10
1433 - 12019	1	(384000 / F) - 12.95

Note: A timed period of silence may be obtained by setting the frequency count value to 0. Specifically, if the frequency - high byte Cx = 0, then the low byte Ct is ignored. The speaker operation is executed exactly as for any valid frequency, except that the frequency signal pin from the 8051 to the speaker is held in its quiescent state.

Speaker Duration Control

A speaker tone duration is specified by the system as the number of duration ticks where each duration tick is 1/128 of a second. Duration ticks are specified as a 15-bit number with a permissible range from 0 to 32,767, X'0000' through X'7FFF'. The actual number of duration ticks is one greater than the number specified, providing a tone duration from 7.8 milliseconds to 256 seconds. System software must first specify the tone duration-high byte, then issue the write to speaker command specifying the duration-low byte.

Speaker Volume Control

The adapter controls the speaker volume for both the adapter's use of the speaker and the Coprocessor feature's use. System software must ensure that only one source is driving the frequency control. In either case, system software must use the adapter function to specify speaker volume.

Four levels of volume are available: off, low, medium, and high. The adapter initializes itself after a reset to medium. System software may change this setting at any time. The adapter retains the last setting in 8051 shared RAM mode bits 8 and 9 as follows:

M9, 8	Volume
B'00'	Off
B'01'	Low
B'10'	Medium (Default)
B'11'	High

Speaker Command Queue Description

Execution of the Write to Speaker command is queued if the speaker facilities are busy doing a click or a previous speaker command. Speaker commands are rejected if a Write to Speaker is currently queued. The queue consists of one set of parameters, frequency and duration. When speaker parameters are written to the adapter, they are initially pending. They become active when speaker facilities are available.

A keystroke click occurs only if speaker facilities are immediately available. The click is not queued.

Speaker volume is a global parameter under user control. As such, it is not queued. Speaker Volume commands are immediately executed.

Speaker Command Queue Operation

Commands to set frequency (high and low bytes) and duration (high byte) place these parameters in the queue as pending. The commands are rejected if the queue is full.

The Write to Speaker command places the duration (low byte) parameter in the queue, with an implied request to activate the speaker using the pending parameters. The command is rejected if the queue is full.

Normal response to Write to Speaker is either speaker started or parameters queued.

Speaker Command Queue Implementation

Speaker command queueing uses the following adapter resources:

- SRAM Address X'01-02' = Pending duration high/low

(Active duration is maintained in SRAM Address X'13-14')

- SRAM Address X'15-16' = Pending frequency high/low

(Active frequency kept in private RAM)

- Status bit S7 defined as speaker queue full
- Status report informational code bit 3: set on if speaker parameters (frequency or duration) are invalid...
  - When a keystroke click occurs (frequency only)
  - When the pending speaker command is dequeued.

Note: Parameters are not validated until actually activated.

Keystroke Click Description

The keyboard does not provide an acoustical feedback for a key being depressed. Instead, the adapter provides a keystroke auto-click function which generates a click tone whenever a valid scan code is received from the keyboard. The resulting clicks only represent a replacement for a mechanically generated keyboard click. The adapter-generated click only means that the adapter has received a valid scan code and queued it to the system. The click cannot be interpreted to mean that the system software has received the keystroke.

Keystroke Click Operation

The keystroke auto-click function defaults to active. It can be disabled by setting mode bit M14 on. The click defaults to a 301 Hz tone for 1.68 milliseconds. The click frequency and duration can be specified by system software.

A keystroke click is initiated only when the adapter receives the make of any valid scan code and queues it to the system. Valid scan codes are those in the range X'01' thru X'9F'.

If the byte received from the keyboard is a break code, then the next byte received will not initiate a click.

If the keystroke queue is full when a keyboard byte is received, the adapter replaces the last byte in the queue with the overrun code X'00', discards the current byte received, and does not initiate a click.

Keystroke Click Implementation

Keystroke auto-click uses the following adapter resources:

- SRAM Address X'07' = Click duration

- SRAM Address X'0B-0C' = Click frequency
- Mode Bit M14 defined as inhibit keystroke auto-click
- Status report informational code bit 3 is set on if the click frequency in SRAM is invalid when the keystroke auto-click is initiated

The click duration is specified by SRAM address X'07' as a count of 30.52 microsecond ticks. The count in SRAM is an 8-bit number. The actual number of duration ticks is one greater than the number specified in SRAM, providing a range of click durations from 30.52 microseconds to 7.8 microseconds in 30.52 microsecond increments.

The click frequency is specified by SRAM addresses X'0B' and X'0C'. These two bytes are used as the frequency counter values Cx and Ct, respectively. They are interpreted by the adapter exactly as for any other speaker frequency. They must adhere to the same rules as specified in the paragraph "Speaker Frequency Control". Cx=0 is invalid, click cannot be the silent tone.

If an invalid click frequency is written to SRAM and the adapter attempts to click because of a keystroke, no click occurs. An unsolicited status report informational code byte with bit 3 on is posted to the system.

Click Suppression for Defined Scan Code Set

The keystroke auto-click function may be suppressed for a definable set of five scan codes by setting mode bit M7 = 1. The default set of scan codes is defined by the five SRAM addresses X '08' through '0A', '0D' and '0E'. (The last two locations also define the system attention sequence.) A scan code may be deleted from the suppress set by setting its corresponding SRAM location to X 'FF'.

Keystroke Click Interference With Other Speaker Operations

No click occurs if the adapter's speaker facilities are busy with a previous speaker command when a click is initiated. Conversely, if the speaker is busy with a click and the system issues a Write to Speaker command, then the parameters for that command are queued and activated when the click completes.

The adapter has no knowledge of the Coprocessor using the speaker. Consequently, if the Coprocessor is using the speaker and the adapter issues a click, the sound observed is a collision of the Coprocessor-generated tone and the click tone.

Adapter RAS and Security Functions

The following are RAS and security functions supported by the adapter.

Keylock Switch Support

When the keylock switch is on, the adapter does the following:

- Inhibit keyboard input
- Inhibit UART device input (from locator)
- Inhibit writes to the keyboard and UART devices.

The adapter conditionally informs the system of changes in the keylock switch setting with a status report informational code (refer to 5-127).

System software can override the keylock switch on by resetting mode bit 13 (M13=0).

Note: When the system is turned on, system ROM code queries the the keylock switch state during initialization and acts appropriately if found on.

Detection of Special Keystroke Sequences

As keystrokes pass from the keyboard to the system through the adapter, the adapter searches for two special sequences of scan codes. One sequence causes the adapter to initiate a system reset and the second sequence initiates a system attention interrupt. The scan codes received by the adapter are always presented to the system with an interrupt ID 1.

Initiate System Reset

The adapter searches incoming keystrokes for the special sequence as follows:

1. Scan Code X'11'
2. Scan Code X'19'
3. Scan Code X'62'

The first two keys must be make/break type.

The first two scan codes may appear in either order. When the sequence is detected, an immediate system reset is initiated. The sequence search is always active (except when the keylock switch is on). However, if system software has disabled the keyboard interface with M11=0, or has issued a command to the keyboard which disables scanning, no keystrokes are received by the adapter and the sequence search is effectively disabled.

Initiate System Attention Interrupt

The adapter searches incoming keystrokes for the special sequence whose default is as follows:

1. Scan Code X'11'
2. Scan Code X'19'
3. Match on any scan code other than X'62'

The first two keys must be make/break type.

When the sequence is detected, an immediate system attention interrupt is initiated. The sequence search can be disabled by system software setting mode bit M6 off. When enabled, the sequence search is always active (except when the keylock switch is on, or if system software has disabled the keyboard interface with M11=0, or has disabled keyboard scanning). The system can alter the keystroke sequence searched, and the length of the sequence can be 1, 2, or 3 keystrokes.

Read/write SRAM defines the three scan codes.

- If the sequence length is 2 or 3, then the first two scan codes may appear in either order.
- If the sequence length is 3 and scan 3 in SRAM = X'FF' (default conditions), then the sequence detection is satisfied when any third scan code is received (other than a break code).
- The third scan code received of a length-3 sequence is always placed in Read-Only SRAM address X'1A' before the system attention interrupt is initiated by the adapter.

Diagnostic Wraps

Certain adapter port signals are diagnostically sensed through the following mechanisms:

- Extended command X'70'
- Reading 8255 PB Port.

Adapter command byte bit 5 = 1 causes the UART TxD signal to be wrapped to the UART RxD on the system board 8051 pins. To wrap a byte, use adapter command Write UART - Control (X'23') with any desired data byte. That data byte will then be read at the UART RxD pin and posted to the system with an interrupt ID 2.

Adapter Self-Test After a Power-On, System, or Adapter Reset

The following 8051 facilities are tested:

- ROM check-sum
- RAM
- Internal registers.

As the tests in this self-test series are executed, a bit-significant completion code is generated. The code is initialized to X'51' and, as each test completes, the corresponding bit is complemented if the test was successful. Tests are executed and bits are complemented from most to least significant with bit meanings as follows:

Bit 7:	Reset initiation indicator	1 = initiated by hardware reset*
Bit 6:	Accumulator and PSW test	0 = test passed ok
Bit 5:	ROM check sum test	1 = test passed ok
Bit 4:	RAM test with AA data	0 = test passed ok
Bit 3:	RAM test with 55 data	1 = test passed ok
Bit 2:	RAM test - addressing	1 = test passed ok
Bit 1:	RAM test with 00 data	1 = test passed ok
Bit 0:	Control regs check sum test	0 = test passed ok

Thus, if all tests pass successfully, the resultant completion code is X'AE'. The completion code is stored in read-only shared RAM address X '1C' and posted to the system in the 8255 with an interrupt ID 6.

* A hardware reset is initiated by a system power-on, system reset, or adapter reset. If bit 7 = 0, then the self-test was initiated by the 8051 having ended abnormally. (Refer to 5-127.)

Diagnose Functions Executed on System Command

- Execute 8051 soft reset (extended command X'60'), force abnormal end code X'A0', perform self-tests, and report the completion code with an interrupt ID 6
- Force system reset (extended command X'61')
- Force system attention interrupt (extended command X'62').

Adapter Informational Codes Returned to System

The data byte associated with interrupt ID 0 is an informational code. Informational codes are classified as:

- Acknowledgment
- Command reject
- Status report.

Acknowledgement Informational Codes

One of the following codes is returned as the data byte associated with interrupt ID 0 to acknowledge receipt of an adapter command from the system.

X'00' = Host cmnd ack
X'01' = Speaker started
X'02' = Speaker inactive
X'03' = Speaker terminated
X'04' = Speaker parameters queued

Command Reject Informational Codes

If the adapter rejects a command from the system, it returns one of the following codes as the data byte associated with Interrupt ID 0.

Reject Code:	Command Handler Issuing:
X'41' = Keyboard transmit busy	Write Keyboard OP
X'42' = Keylock active	Write Keyboard OP, and Write UART OP
X'43' = Keyboard disabled	Write Keyboard OP
X'44' = Invalid keyboard data	Write Keyboard OP
X'47' = Invalid speaker duration	Write Speaker OP
X'48' = Invalid speaker freq	Write Speaker OP
X'4A' = Speaker queue full	Write Speaker OP
X'4B' = UART disabled	Write UART OP
X'4C' = UART transmit busy	Write UART OP
X'4D' = Invalid baud	Set UART Baud Rate OP

X'4E' = Invalid framing	Initialize UART Framing OP
X'50' = Invalid count	Set Sequence a Length OP
X'51' = Illegal mode	Diagnose OP, and Diagnostic Sense OP
X'60' = RAM queue busy	Dump RAM Block OP
X'7F' = Undefined op	

Status Report Informational Codes

The adapter may send an unsolicited status report informational code to the system with interrupt ID 0. A status report is distinguished from acknowledgement and command reject codes by having the most significant bit of the byte set on. The remaining bits of the status report byte are bit-significant. The bits are defined as follows:

Bit 7 = 1 (Status Report Identifier)	
Bit 6 = Speaker tone completed	conditioned by M3=1
Bit 5 = Keyboard returned ack	conditioned by M0=1
Bit 4 = Keylock switch changed	conditioned by M2=1
Bit 3 = Invalid speaker parameter	(click or queued freq or dura)
Bit 2 = UART transmit complete	conditioned by M1=1
Bit 1 = RAS log near overflow	
Bit 0 = RAS log overflowed.	

Adapter Error Codes Returned to System

The data byte associated with interrupt ID 7 is an error code. Error codes are classified as:

- Abnormal end codes
- Device error codes.

Abnormal End Codes

Explanation

The adapter has detected an unrecoverable error condition and terminated normal operations. The data byte (abnormal end code) provides the specific condition detected. When the abnormal end code has been read by the system, two additional bytes are provided by the adapter with an interrupt ID 7 which define the 8051 microcode address which detected the condition. After the microcode address has been transferred to the system, or if the 8051 times out waiting for the system, the 8051 will re-initialize itself and attempt to restore normal operations. The 8051 shared RAM is reset to its default state. After the soft reset self-test has been performed by the 8051, the completion code is reported with an interrupt ID 6. The high order bit of the completion code will be off, indicating a soft reset of the 8051 rather than a hardware initiated reset.

System Action Required

When the system software detects an abnormal end code, it should prepare to receive the subsequent two codes (8051 address). That information should be logged as an incident. Re-initialize the 8051 shared RAM, if any of the defaults had been previously altered. If an abnormal end code immediately re-occurs, the system should issue an adapter reset operation.

Codes

X'A0' = Diagnose initiated 8051 Soft Reset
X'A1' = Work queue low decode
X'A3' = Host transmit queue decode
X'A4' = Incr RAS log decode
X'A6' = Wild branch
X'A7' = System reset failed

Device Error Codes

Explanation

An unexpected condition has been detected by the 8051 microcode at a device interface. The problem may not be with the device itself. The Adapter will attempt to continue normal operations.

System Action Required

The device error code information should be logged as an incident. The system may have to issue some kind of device reset command to try and clear the condition. A particular device error code may suggest a recovery procedure. If the condition persists, the system should issue an adapter reset operation.

Codes

X'E0' = Keyboard Transmit Timeout

Adapter has started to transmit a frame to the keyboard. Transmission of that frame exceeded the maximum allowed time. The keyboard interface has been cleared and re-enabled. The keyboard Echo command could be issued to test the circuits to the keyboard and back. If the condition continues, the keyboard may have been unplugged.

X'E1' = Keyboard Receive Timeout

The adapter has started to receive a frame from the keyboard. Reception of that frame exceeded the maximum allowed time. The keyboard interface has been cleared and re-enabled if Mode bit 11 = 1. The keyboard Echo command could be issued to test the circuits to the keyboard and back. If the condition continues, the keyboard may have been unplugged.

X'E2' = Kbd Ack Not Received

An acknowledgement response was expected from the last transmission to the keyboard, but something other than an acknowledgement was received. The actual keyboard response byte has been queued to the system. The most likely problem is that the keyboard rejected the command or parameter byte sent to it.

X'E3' = Unexpected Kbd Ack Received

An unexpected acknowledgement response was received from the keyboard. It was unexpected because a prior transmission had not been performed by the adapter.

X'E4' = Hard Error on Kbd Frame Receive

The adapter has unsuccessfully performed the maximum number of keyboard frame receive retries. The frame received has a solid framing error. The keyboard interface is enabled for further communication.

X'E5' = Hard Error on Kbd Frame Transmit

The adapter has unsuccessfully performed the maximum number of keyboard frame transmit retries. The keyboard has responded with a solid error condition (resend) because of the keyboard receiving an invalid frame. The keyboard interface is enabled for further communication.

X'E6' = Kbd Clock Pin Not Plus

(See following E7 explanation)

X'E7' = Kbd Clock Pin Not Minus

The adapter has attempted to release or hold the clock signal to the keyboard. The read-back of the clock line signal did not verify. If this condition persists, it may suggest a failed interface circuit. It may also be due to spurious noise on the cable, or the keyboard may have been unplugged. The adapter attempts to continue the operation.

X'E8' = UART Interrupt Without TIRI

A serial port interrupt occurred without a transmit or receive identifier. The interrupt is ignored and processing continues.

X'E9' = UART Transmit Timeout

The 8051 serial port buffer was loaded for transmission to the UART device. Transmission exceeded the maximum allowed time. Processing continues. A UART device wrap command could be issued. If the condition continues, the UART device may have been unplugged, or the 8051 serial port is bad.

X'EA' = UART Ack Timeout

The 8051 serial port transmitted the byte to the UART device. The device did not respond within the maximum allowed time. Processing continues. Other device commands could be issued. If no response, the device may have been unplugged.

Adapter Device Support Notes

The information in this section supplements that in the respective keyboard and locator sections of this manual.

Keyboard Commands

Resend (X'FE')

The system should not issue this command. The function is handled internally by the adapter's retry facility.

Echo (X'EE')

The adapter passes the keyboard response to the system.

Keyboard Outputs

Resend (X'FE')

The adapter intercepts this output and handles it in the adapter's retry facility. The X'FE' is not passed on to the system.

Command Reject (X'FC')

The adapter posts the X'E2' device error code and the byte received from the keyboard (X'FC') to the system.

Acknowledgment (X'FA')

The adapter intercepts this output and reports the occurrence of the acknowledgment only if mode bit 0 = 1.

Overrun (X'00)

The adapter also returns an overrun (X'00') if keystrokes fill the adapter's 5-byte FIFO queue.

Locator Device Support Notes

Adapter Command: Write UART - Control (X'03')

All UART device commands not having a defined response must be issued with the Write UART - Control command to the adapter. This includes the following mouse commands:

- Enable
- Disable
- Set wrap mode
- Reset wrap mode
- Reset exponential scaling
- Set sampling speed
- Set incremental stream mode
- Set remote data mode.

Adapter Command: Write UART - Query (X'04')

All UART device commands having a defined response must be issued with the Write UART - Query adapter command.

The following mouse commands, with the required mode bit 5 setting, should be issued with the Write UART - Query adapter command:

- Reset (M5=1)
- Read configuration (M5=0)
- Read X, Y data (M5=1)
- Query status (M5=1).

The blocking factor in shared RAM address X'19' must define a 4-byte report for the mouse.

When the mouse is in wrap mode, subsequent bytes to be wrapped to the mouse must be written to the adapter using Write UART - Query with M5=0. While in wrap mode, the mouse accepts any parity value and returns the parity bit to the adapter as received. The system can control how the adapter generates parity sent to the mouse (and checks the parity returned from the Mouse) by setting shared RAM address X'19' (MSB) appropriately.

Adapter Design Notes

Figure 5-57 on page 5-133 shows the adapter components and the system interface. The IBM 6151 and IBM 6150 use a 9.216 MHz oscillator to drive the 8051 oscillator input.

Figure 5-58 on page 5-134 shows the interface logic for keyboard and speaker signals between the 8051 and the connector to the keyboard. (The speaker is mounted within the keyboard covers.)

Figure 5-59 on page 5-135 shows the interface logic for locator signals between the 8051 and the connector to the locator.

Also shown in the interface figures are the points where signals are wrapped to the 8255 Port B inputs for diagnostic sensing.

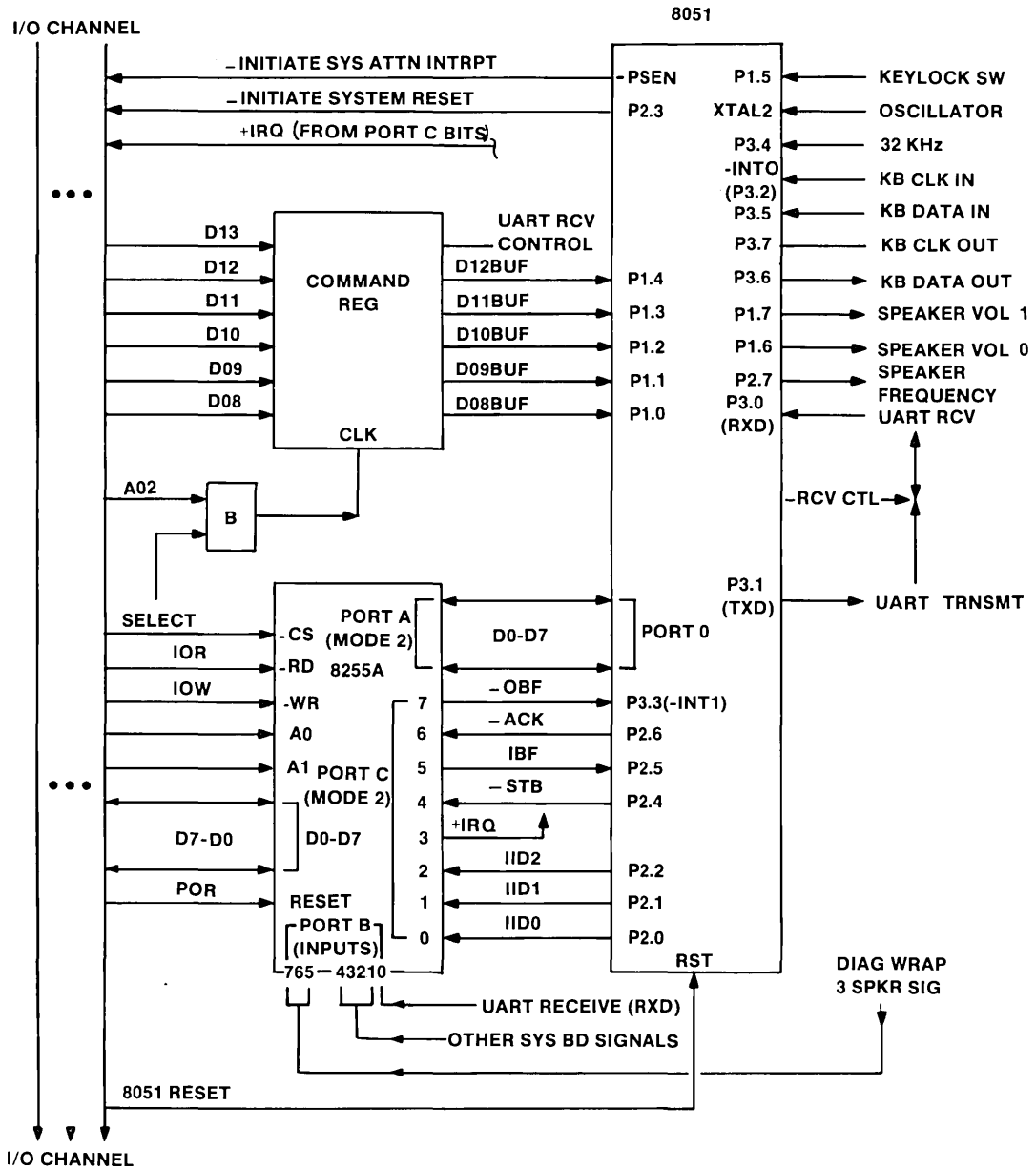


Figure 5-57. Adapter Logic and System Interface

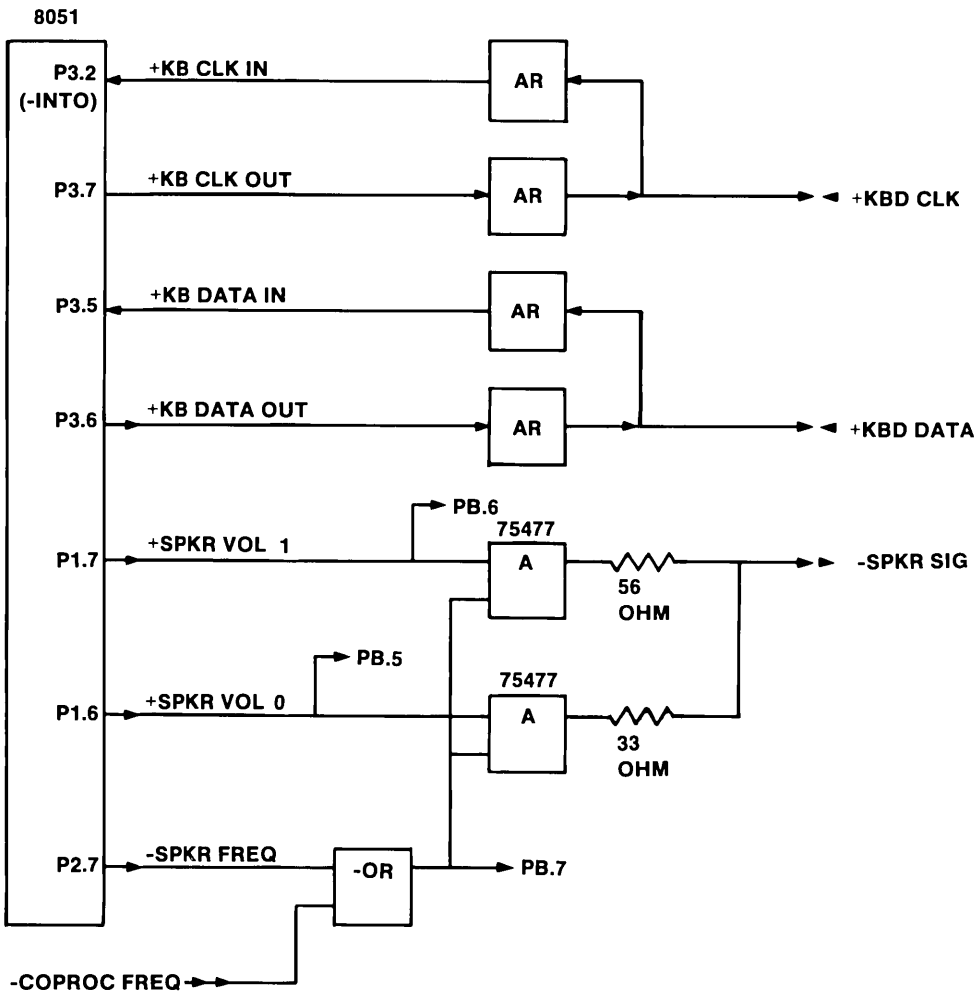


Figure 5-58. Adapter-to-Keyboard Connector Interface

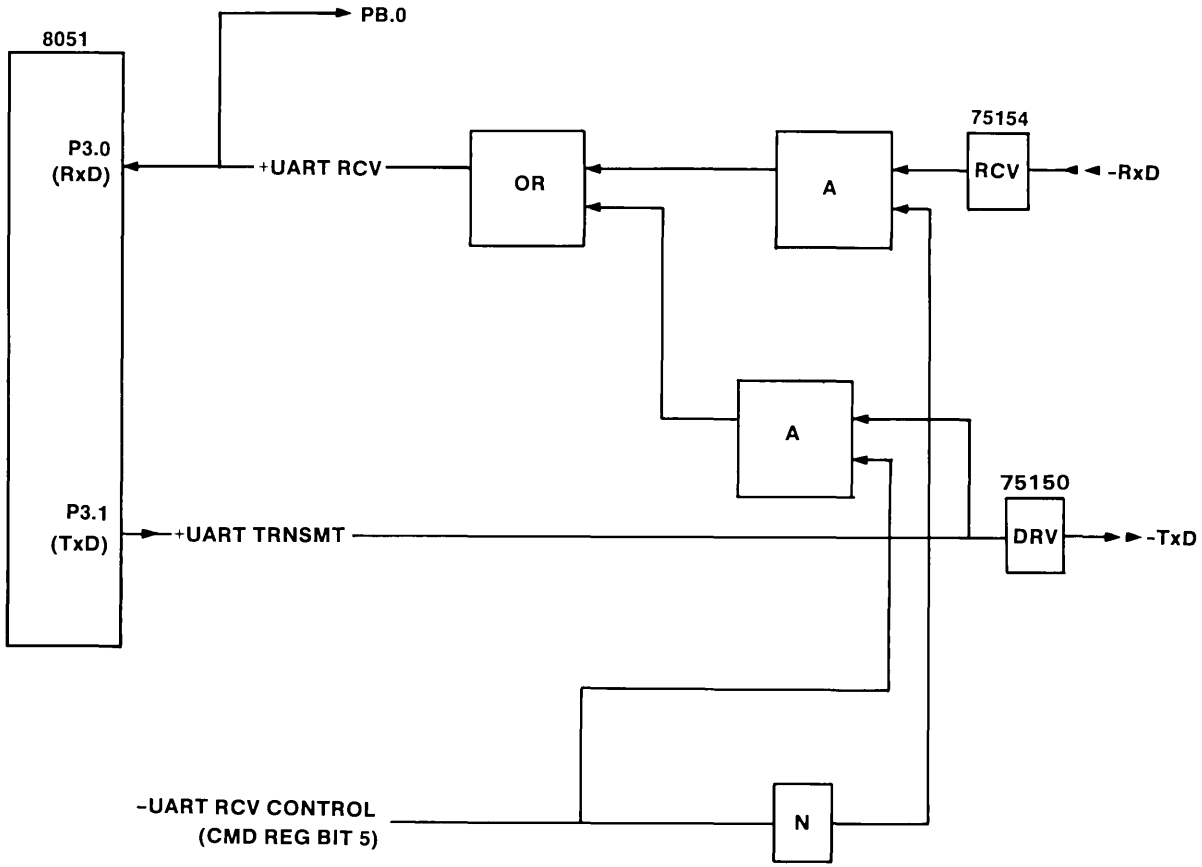


Figure 5-59. Adapter-to-Locator Connector Interface

8051 Pin Assignment

Pin Number	Port/ Bit	In/ Out	8051 Function	Adapter Function
39	0.0	B		+I/O buffer bit 0
38	0.1	B		+I/O buffer bit 1
37	0.2	B		+I/O buffer bit 2
36	0.3	B		+I/O buffer bit 3
35	0.4	B		+I/O buffer bit 4
34	0.5	B		+I/O buffer bit 5
33	0.6	B		+I/O buffer bit 6
32	0.7	B		+I/O buffer bit 7
1	1.0	I		+System data bus bit 8 buffered
2	1.1	I		+System data bus bit 9 buffered
3	1.2	I		+System data bus bit 10 buffered
4	1.3	I		+System data bus bit 11 buffered
5	1.4	I		+System data bus bit 12 buffered
6	1.5	I		+Keylock switch
7	1.6	O		+Speaker volume bit 0
8	1.7	O		+Speaker volume bit 1
21	2.0	O		+8255 PC bit 0 = interrupt ID bit 0
22	2.1	O		+8255 PC bit 1 = interrupt ID bit 1
23	2.2	O		+8255 PC bit 2 = interrupt ID bit 2
24	2.3	O		-Initiate system reset
25	2.4	O		-Strobe to input buffer
26	2.5	I		+Input buffer full
27	2.6	O		-Gate output buffer to 8051
28	2.7	O		+Speaker frequency
10	3.0	I	RXD	UART (locator) receive data bit
11	3.1	O	TXD	UART (locator) transmit data bit
12	3.2	I	-INT0	+Keyboard clock in

Pin Number	Port/ Bit	In/ Out	8051 Function	Adapter Function
13	3.3	I	-INT1	-Output buffer full
14	3.4	I	T0	32,768 Hz clock
15	3.5	I	(T1)	+Keyboard data in
16	3.6	O	(WR)	+Keyboard data out
17	3.7	O	(RD)	+Keyboard clock out
9		I	RST	+8051 reset
18		I	XTAL2	Oscillator
19		I	XTAL1	
20		P	VSS	Ground
29		O	(PSEN)	-Initiate system attention interrupt
30		O	ALE	
31		I	-EA	+5V
40		P	VCC	+5V

8255 Pin Assignment

Pin Number	Port/ Bit	In/ Out	8255 Function	Adapter Connection
34		I	D0	+System data bus bit 0
33		I	D1	+System data bus bit 1
32		I	D2	+System data bus bit 2
31		I	D3	+System data bus bit 3
30		I	D4	+System data bus bit 4
29		I	D5	+System data bus bit 5
28		I	D6	+System data bus bit 6
27		I	D7	+System data bus bit 7

Pin Number	Port/ Bit	In/ Out	8255 Function	Adapter Connection
4	PA.0	B		+I/O buffer bit 0
3	PA.1	B		+I/O buffer bit 1
2	PA.2	B		+I/O buffer bit 2
1	PA.3	B		+I/O buffer bit 3
40	PA.4	B		+I/O buffer bit 4
39	PA.5	B		+I/O buffer bit 5
38	PA.6	B		+I/O buffer bit 6
37	PA.7	B		+I/O buffer bit 7
18	PB.0	I		+UART receive (RxD)
19	PB.1	I		+TXDA (system BD serial port A)
20	PB.2	I		+RXDA (system BD serial port A)
21	PB.3	I		-Timer in (system BD signal)
22	PB.4	I		-Interrupt 1 (system BD signal)
23	PB.5	I		+Speaker volume bit 0
24	PB.6	I		+Speaker volume bit 1
25	PB.7	I		+CPSpeaker/speaker frequency
14	PC.0	I		+Interrupt ID bit 0
15	PC.1	I		+Interrupt ID bit 1
16	PC.2	I		+Interrupt ID bit 2
17	PC.3	O	INTR	+IRQ
13	PC.4	I	-STB	-Strobe 8051 data to input buffer
12	PC.5	O	IBF	+Input buffer full
11	PC.6	I	-ACK	-Gate output buffer to 8051
10	PC.7	O	-OBF	-Output buffer full
6		I	-CS	-Keyboard chip select
5		I	-RD	-IOR
36		I	-WR	-IOW
9		I	A0	+Channel address bit 13
8		I	A1	+Channel address bit 14
7		P	GND	Ground

Pin Number	Port/ Bit	In/ Out	8255 Function	Adapter Connection
26		P	VCC	+5V
35		I	RST	+Power on reset

Channel I/O Device Address Bit Decoding

Read/Write I/O Subsystem Keyboard Adapter

```
                  MSB                 LSB
                    1111 11           |
                    5432 1098 7654 3210
                    -------------------
I/O Address:        1000 0100 0000 ORPP        = X'840-'

Where:          PP = 8255 PORT ADDRESSING FOR I/O READ & WRITE
                     PP   IOR               IOW
                     --   ------------      ------------
                     00:  Read PA Reg       Write PA Reg
                     01:  Read PB Reg       Write PB Reg
                     10:  Read PC Reg       Write PC Reg
                     11:  Illegal           Write Control

                     PA is defined to be the adapter data reg.
                     It will send/receive the low-order data byte.

                R =  Adapter command register control
                     R    IOR               IOW
                     --   -----------       ------------
                     1:   No Action         No Action
                     0:   No Action         Latch High Byte in
                                             Adapter Register
```

Note: A normal 2-byte write operation to the adapter would have RPP = 000, to I/O address X'8400'. This loads the command and data bytes in their respective registers.

Selective Reset I/O Subsystem Keyboard Adapter

```
                    MSB                 LSB
                      1111 11           |
                      5432 1098 7654 3210
                      -------------------
I/O Address:          1000 1100 0110 0000   = X'8C60'
```

An I/O write to address X'8C60' will set or reset the 8051 selective reset latch. Data bit 2 determines whether the latch is set or reset . . .

- SET: Data bit=0
- RST: Data bit=1

When the latch is set, the 8051 is held in a reset condition; when the latch is reset, the 8051 performs its self-test and initialization.

Note: The 8255 is only reset with an actual power-on.

8051 RAM Allocation

Absolute RAM Address	Shared RAM Address	Usage	Number of Bytes
00-07		Reg Bank 0	8
08-0F		Reg Bank 1	8
10-17		Reg Bank 2	8
18-1F		Reg Bank 3	8
20-2B		Work Area	12
2C-2E	10-12	24-Bit Shared Bit Space	3
2F-3B	13-1F	Shared (R.O.) Byte Space	13
3C-4B	00-0F	Shared (R/W) Byte Space	16
4C-57	20-2B	RAS Logs	12
58-63		UART Current Receive Blocks	12

Absolute RAM Address	Shared RAM Address	Usage	Number of Bytes
64-68		Keystroke Queue	5
69-7F		Stack	23
		Total	128

Adapter and Keyboard Initialization Procedure

The following are the recommended steps for initializing the adapter and keyboard after a system reset occurs.

1. Activate Adapter Reset

   The 8051 reset is activated by power on, system reset, or system software directly activating the bit in the CRRB register. Additionally, power on causes a POR to the 8255.

2. Configure 8255

   Write the adapter operation config 8255 to properly configure the 8255 for communication with the 8051.

3. Enable Host Interrupt IRQ

   Write the adapter operation enable IRQ to allow the interrupt request line to the system processor to be activated when the adapter is initiating a transfer to the system.

   Note: This step may be done at some later time if interrupts cannot be handled yet. Data transfers in subsequent steps will be either interrupt-driven, or polled.

4. Variable Delay Td

   The variable delay td allows the keyboard to execute its power-on reset tests, and quiesce clock and data lines before the 8051 interface to the keyboard is active. The delay Td is determined by the relation:

   ```
   Tkp + Td >= Tkg + Tkb + Tkc
   ```

   or,

   ```
   Td >= (Tkg - Tkp) + Tkb + Tkc
   ```

   Where:

   ```
   Tkp = time since keyboard power applied
         (usually same as time since system POR)

   Tkg = time for keyboard to power up
       = 2 seconds (maximum) per keyboard specification

   Tkb = time for keyboard BAT to run
       = 300 msec to 500 msec per keyboard specification

   Tkc = time for keyboard to transmit completion
       < 20 msec
   ```

   If the time since system POR is greater than about 2.5 seconds, then no delay is necessary.

5. Release Adapter Reset

   Releasing 8051 Reset in CRRB allows the 8051 to run its self-test and initialization.

6. Wait for Adapter Initialization Response

   After the 8051 self-test and initialization is completed, the completion code is posted to the 8255. This should occur within 100 msec of the release above.

7. Validate Adapter Self-Test Completion Code

   The expected GOOD MACHINE data value is X'AE' with interrupt ID 6.

8. Reset the Keyboard

   Issue the adapter command to reset the keyboard. The expected adapter command response is X'00' with interrupt ID 0. If the response is X'42', the keylock switch is on and the adapter cannot write to or receive from the keyboard.

9. Wait for Keyboard Initialization Response

   Wait for 3 data bytes from the keyboard, which should be received within 600 msec.

   Note: If the first or only byte returned is a device error code (X'E0'), the keyboard is probably unplugged.

10. Validate Keyboard Reset

All data bytes should have been returned with an ID 1. Data bytes should be as follows:

Keyboard BAT completion code	=	X'AA'
Keyboard ID - 1st byte	=	X'BF'
Keyboard ID - 2nd byte	=	X'Bt'

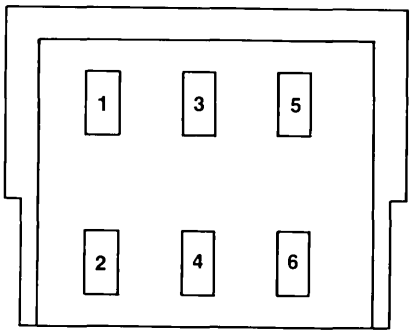
Where: t = keyboard type (see keyboard specification for valid types).

The adapter and keyboard are now initialized to their defined defaults.

System Board Connectors

Keyboard Connector

The following figure shows the keyboard connector (receptacle) on the system board.

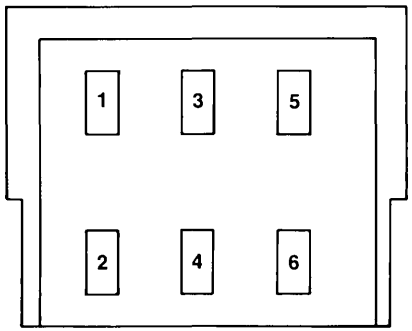


PIN	SIGNAL
1	+5 VOLTS
2	KEYBOARD DATA
3	SPEAKER SIGNAL
4	SPEAKER RETURN (+5V)
5	GROUND
6	KEYBOARD CLOCK

Figure 5-60. Keyboard Connector

Locator Device Connector

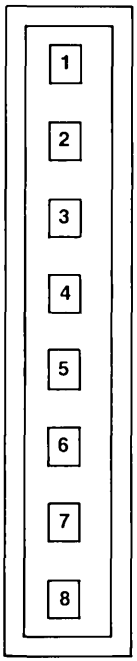
The following figure shows the locator connector (receptacle) on the system board.



PIN	SIGNAL
1	GROUND
2	TRANSMIT TO DEVICE
3	+12 V
4	-12 V
5	+5 V
6	RECEIVE FROM DEVICE

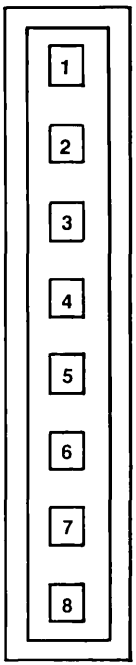
Figure 5-61. Locator Connector

IBM 6150 Power Supply Connectors



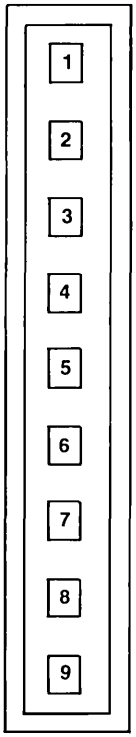
PIN	SIGNAL
1	+5 V
2	+5 V
3	+5 V
4	+5 V
5	GROUND
6	GROUND
7	GROUND
8	GROUND

Figure 5-62. IBM 6150 Power Supply Connector (J1)



PIN	SIGNAL
1	+5 V
2	+5 V
3	+5 V
4	+5 V
5	GROUND
6	GROUND
7	GROUND
8	GROUND

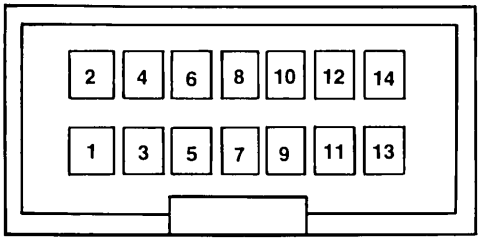
Figure 5-63. IBM 6150 Power Supply Connector (J2)



PIN	SIGNAL
1	EPOW
2	NO CONNECTION
3	POR
4	-12 V
5	+12 V
6	-5 V
7	GROUND
8	GROUND
9	GROUND

Figure 5-64. IBM 6150 Power Supply Signal Connector (J3)

IBM 6150 Operator Panel Connector



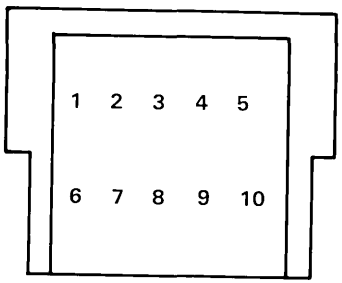
PIN	SIGNAL
1	DATA 06
2	DATA 04
3	GROUND
4	+5 V
5	DATA 05
6	DATA 07
7	GROUND
8	DATA 02
9	DATA 01
10	DATA 03
11	POWER GOOD
12	DATA 00
13	KEYLOCK
14	BATTERY BACKUP

Figure 5-65. IBM 6150 Operator Panel Connector

IBM 6150 Battery Connector

The IBM 6150 battery connector is not located on the system board. The battery plugs into the operator panel for easy access. See "IBM 6150 Battery Connector" on page 9-32.

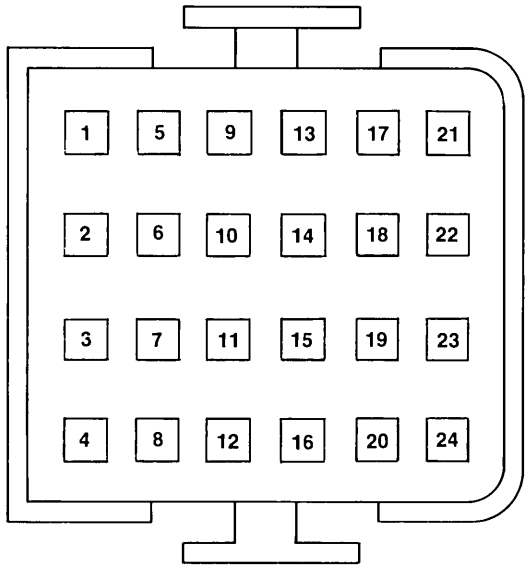
IBM 6150 Serial Port Connectors



PIN	SIGNAL
1	Transmit Data
2	Data Terminal Ready
3	Request to Send
4	Ring Indicate
5	No Connection
6	Receive Data
7	Data Set Ready
8	Clear to Send
9	Data Carrier
10	Signal Ground

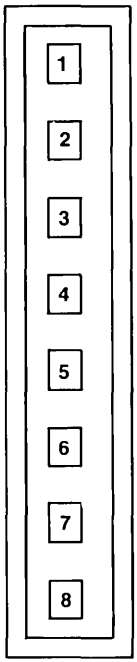
Figure 5-66. IBM 6150 Serial Port Connectors

IBM 6151 Power Supply Connectors



PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL
1	+5 V	2	+5 V	3	+5 V	4	+5 V
5	+5 V	6	+5 V	7	+5 V	8	+5 V
9	+12 V	10	-5 V	11	NC	12	NC
13	-12 V	14	NC	15	NC	16	NC
17	GND	18	GND	19	GND	20	GND
21	GND	22	GND	23	GND	24	GND

Figure 5-67. IBM 6151 Power Supply Connector (J1)

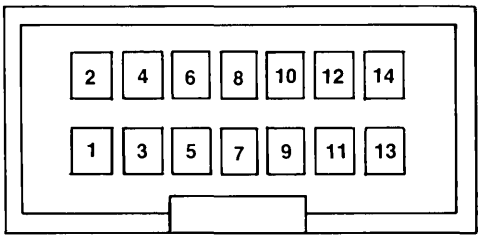


PIN	SIGNAL
1	GROUND
2	KEY
3	NO CONNECTION
4	POR
5	EPOW
6	POWER GOOD
7	NO CONNECTION
8	RESERVED

TR 249

Figure 5-68. IBM 6151 Power Supply Signal Connector (J3)

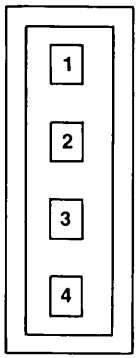
IBM 6151 Operator Panel Connector



PIN	SIGNAL
1	DATA 06
2	DATA 04
3	GROUND
4	+5 V
5	DATA 05
6	DATA 07
7	GROUND
8	DATA 02
9	DATA 01
10	DATA 03
11	POWER GOOD
12	DATA 00
13	KEYLOCK
14	BATTERY BACKUP

Figure 5-69. IBM 6151 Operator Panel Connector

IBM 6151 Battery Connector



PIN	SIGNAL
1	POWER FROM BATTERY
2	KEY
3	NO CONNECTION
4	GROUND

Figure 5-70. IBM 6151 Battery Connector

Section 6. I/O Channel

CONTENTS

About this Section

This section describes the architecture of the I/O channel in detail. It provides information about the I/O channel signals and describes the channel operations that are required to transfer data.

Channel Features

The RT PC I/O Channel features a 62-pin I/O connector that accommodates some IBM PC adapters. It also has a 40-pin connector that accommodates some PC-AT adapters. There are two additional +5Vdc and two additional ground pins on the 40-pin connector to support high wattage adapters.

- A 16M-byte memory address space
- A 64K-byte I/O address space
- A 16-bit data bus
  - Supports 8 or 16 bit devices
- A multiple controller arbitration
  - 8 DMA channels
- Eleven maskable interrupts
- A nonmaskable interrupt

I/O Channel Conventions

Signal Levels

Signals are high-active unless the signal name is preceded by the symbol "-", which implies that the signal is low-active.

Most and Least Significant Bits

		MSB	LSB
ADDRESS	PC-1, PC-XT	SA19	SA0
	RT PC, PC-AT	LA23	SA0
DATA	PC-1, PC-XT	SD7	SD0
	RT PC, PC-AT	SD15	SD0

Note: Only those adapters identified in this manual are usable in the RT PC system.

Byte Alignment

1 Byte = 8 Bits
1 Word = 16 Bits = 2 Bytes
1 Double Word = 32 Bits = 4 Bytes

- Byte Organization

Byte Address	SAO	-SBHE	Bits 7-0
000000	0	1	Byte 0
000001	1	0	Byte 1
000002	0	1	Byte 2

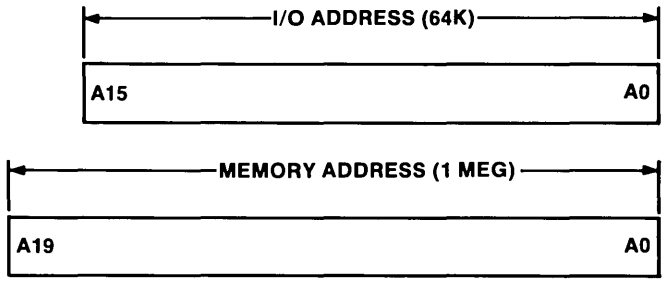
- Word Organization

Even byte data (SAO = 0) is carried on SDO-SD7.

Odd byte data (SAO = 1) is carried on SD8-SD15.

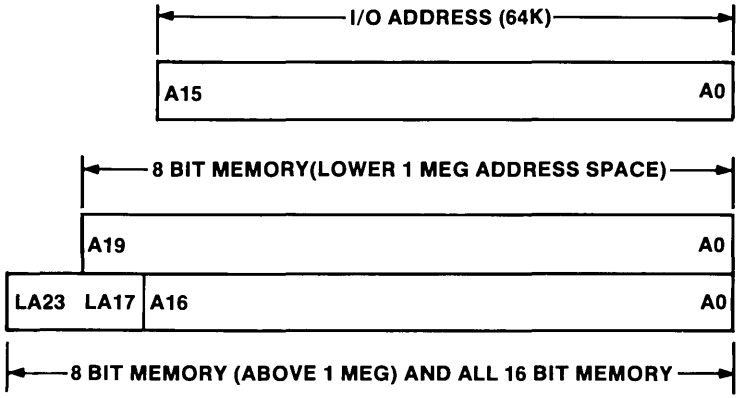
Word Address	SA1	SAO	-SBHE	Bits 15-8	Bits 7-0
000000	0	0	0	Byte 1	Byte 0
000002	1	0	0	Byte 3	Byte 2

Addressing Scheme



Device Addressed	Bit Assignments	Address Space
I/O Adapters	SA0-SA15	64K I/O Address Space
Memory	SA0-SA19	1 MEG Memory Space

Figure 6-1. 62-Pin PC Adapters



Device Addressed	Bit Assignments	Address Space
I/O Adapters	SA0-SA15	64K I/O Address Space
8 Bit Memory	SA0-SA19 SA0-SA16 and LA17-LA23	0-1 MEG Memory Space 1-16 MEG Memory Space
16 Bit Memory	SA0-SA16 and LA17-LA23	0-16 MEG Memory Space

Figure 6-2. 98 and 102 Pin PC-AT or RT PC Adapters

I/O Channel Signal Definitions

Note: See "I/O Channel Data Transfer" on page 6-15 for definition of terms.

- SA0 - SA19

  System address bits SA0-SA19 are used by the current DMA controller to address system memory or any other device, I/O adapter or memory on the I/O channel. The DMA Controller must hold these address lines stable during the entire data transfer cycle.

  During I/O cycles address bits SA16-SA19 will be B'0000' and address bits SA0-SA15 will be used to access a 64K I/O space. During memory cycles all 20 address bits will be used to access the low 1 megabyte of memory address space.

  SA17-SA19 must be received by 8-bit memories and they must be driven by any DMA Controller which must transfer data with them.

- LA17 - LA23

  I/O channel address bits LA17-LA23 are used with SA0-SA16 to extend the I/O Channel memory address space to 16 megabytes. Sixteen bit memory devices must receive LA17-LA23, and all DMA Controllers must drive them. During I/O cycles the LA address bits are all 0.

  The DMA controller must hold these address lines stable during the entire data transfer cycle.

  Note: The RT PC system board drives all (SA0-SA19 and LA17-LA23) address signals through the complete channel cycle. 'BALE' is not driven by the RT PC system board. However, the line is implemented for adapters that need it.

  Application Note: Some system boards *may not* latch these signals and they *do not remain valid* through the complete channel cycle. Any device receiving these signals must use 'BALE' to latch these I/O channel addresses. A transparent latch (such as 74ALS573) is required. (See BALE.)

- BALE

  Address latch enable is used to latch valid addresses from a system board or DMA controller that does not latch addresses LA17-LA23. These addresses are valid and should be latched on the falling edge of 'BALE'.

  For all DMA controller cycles the system board IOCC will drive 'BALE' high, allowing addresses LA17-LA23 to pass-through the transparent latch. (See Application Note on page 6-8.)

- -SBHE

  Bus high enable indicates a data transfer on the high byte of the 16 bit data bus (SD8-SD15) and is used with SA0 to distinguish between odd byte, even byte and word transfers. (See Figure 6-3 on page 6-10.)

- AEN

  Address enable is active (high) on the I/O channel during all DMA device cycles on the I/O channel. For all other alternate controller cycles the IOCC will drive it (low).

  Note: 'AEN' must be used as part of the device select decode for all I/O space cycles. Since '-IOR' or '-IOW' is active during DMA adapter operations (which is a memory space operation), 'AEN' must prevent false selects during these cycles. I/O space selects should be enabled only when 'AEN' is low.

- SD0 - SD15

  The system data lines transfer data between adapters or options and memory. The DMA adapter sends data on write cycles and the memory sends data on read cycles. Refer to Figure 6-3 on page 6-10 for details of transfers between 8 and 16 bit devices.

Note: 8-bit devices must attach to the low order byte (SD0-SD7) on the channel.

Transfer Type	-SBHE	SA0		SD8-SD15		SD0-SD7	
8-bit DMA Device *3 and	1	0		X		Even	
8-bit Device	0	1		X		Odd	
8-bit DMA Device *3 and	1	0		X		Even	
16-bit Device	0	1	(Read)	Odd	—>	T	*1
	0	1	(Write)	T	<—	Odd	*1
16-bit DMA Controller *4 and	0	0		X		Even	
8-bit Device	0	1	(Read)	T	<—	Odd	*2
	0	1	(Write)	Odd	—>	T	*2
	1	0		X		Even	
	1	1		T		T	
16-bit DMA Controller and	0	0		Odd		Even	
16-bit Device	0	1		Odd		T	
	1	0		T		Even	
	1	1		T		T	

Figure 6-3. Data Transfer Table

Note: The arrow shows the direction SD8-SD15 are multiplexed.

Even **Even addressed byte**
Odd **Odd addressed byte**

T Tri-state
X Don't care
*1 The IOCC must multiplex odd byte data for transfers between a 16 bit channel attached device and an 8-bit DMA device.
*2 The IOCC will also multiplex odd byte data during transfers between alternate controllers and 8-bit channel-attached devices.
*3 DMA devices only
*4 Alternate controllers only

Note: The IOCC handles all multiplexing of data to and from system memory for alternate controllers and DMA device operations.

- -SMEMR; -MEMR

  The Memory Read Command instructs the selected memory device in the memory address space to send data onto the data bus. '-SMEMR' is active only when the memory decode is within the low 1 megabyte of the I/O memory space. '-MEMR' is active on all memory read cycles. The adapter option must hold data on the channel when memory read is active.

- -SMEMW; -MEMW

  The Memory Write Command instructs the selected memory device in the memory address space to store the data present on the data bus. '-SMEMW' is active only when the memory decode is within the low 1 megabyte of the I/O channel memory space. '-MEMW' is active on all memory write cycles.

  Note: '-MEMW and -MEMR' should be used by all 16-bit devices. '-SMEMW and SMEMR' (along with SA17-SA19) must be driven by any controller which must transfer data with devices operating in the low 1 megabyte of memory (such as system board, Coprocessor, and IPL devices). LA17-LA19 must equal SA17-SA19, and LA20-LA23 must be 0.

  Note: '-SMEMR, -MEMR, -SMEMW and -MEMW' will not be active on the I/O channel when the target memory is translated to system memory. See "Translation Control Words" on page 5-27.

- -IOR

  The I/O Read Command instructs the selected device in the I/O address space or the 'DACK-selected' DMA adapter to send data onto the data bus. The selected device must hold data on the data bus when I/O read is active.

- -IOW

  The I/O Write Command instructs the selected device in the I/O address space or the 'DACK-selected' DMA adapter to store the data present on the data bus.

  Note: DMA devices and I/O devices which use default timing cycles must use the trailing edge of '-IOW' to sample data to guarantee valid data setup time.

- I/O CH RDY

  I/O Channel Ready (normally high) is pulled low by a selected device to lengthen the current I/O or memory cycle. Any slow device using this line should drive it low when a valid select is decoded and a read or write command is active. Once the device has stored data during a write operation or it has gated valid data onto the data bus during a read operation, it should then drive 'I/O CH RDY' high to indicate that the controller may end the current operation.

  Notes:

  1. 'I/O CH RDY' should not be held not ready by the selected device longer than 2.1 microseconds. The only exception is the system board in device mode may take longer for certain system memory operations.
  2. For devices which store data on the trailing edge of the write command, 'I/O CH RDY' is used to guarantee the data set-up time.

- -MEMCS16

  This signal must be driven active by a 16-bit memory space device after decoding a valid address select using LA17-LA23 only. To meet timing requirements '-MEMCS16' *must not wait* for the latched addresses (such as SA0-SA16 and BALE). A fast default cycle time (see "I/O Channel Cycle Default Timings" on page 6-18) is used by the DMA controller, unless 'I/O CH RDY' is not ready.

- -I/OCS16

  This signal must be driven active by a 16-bit I/O space device after decoding a valid address select from SA0-SA15. A fast default cycle time (see "I/O Channel Cycle Default Timings" on page 6-18) is used by the DMA controller, unless 'I/O CH RDY' is not ready.

  Notes:

  1. It is possible that '-I/OCS16' will be active during a memory cycle or that '-MEMCS16' will be active during an I/O cycle. The DMA controller must know whether an I/O or memory cycle is active and only look at the appropriate signal.
  2. A 16-bit device may use 'I/O CH RDY' to override the fast default cycle time of either '-MEMCS16 and -I/OCS16' if it needs to run slower.
  3. The IOCC will use '-MEMCS16 and -I/OCS16' to distinguish between 8 and 16-bit devices. See "Multiplexing Rules in Figure 6-3 on page 6-10 ." A 16-bit DMA controller can use '-MEMCS16 AND -I/OCS16' to enable a fast default channel cycle.

- DRQ0-DRQ3, DRQ5-DRQ7

  The DMA request lines are used by DMA devices or alternate controllers to request control of the I/O channel from the system arbiter. The requests are ranked with 'DRQ0' having the highest priority and 'DRQ7' the lowest priority. A request is generated by driving a'DRQ' line high. A 'DRQ' line is held high until the corresponding '-DACK' line goes active for DMA devices. For alternate controllers, the 'DRQ' line must be held active for the complete duration of the cycle.

  'DRQ0-DRQ3' are used for 8-bit DMA devices and 16-bit alternate controllers. 'DRQ5-DRQ7' are used by 16-bit DMA devices and alternate controllers. All alternate controllers are 16-bit devices, and they may use any of the 'DRQ' lines. For more details on arbitration, see "Arbitration" on page 5-33 and "DMA" on page 5-39.

  Note: The 'DRQ' lines must be driven with a tri-state driver, which can be enabled or disabled by a software output command. DRQ's must be disabled by system reset (RESET DVR). Since a tri-state level is indeterminate, the system DRQ must be disabled whenever an adapter's DMA request line (DRQ) is tri-stated. There must be a 60 nanosecond wait time from disabling one tri-state driver before turning on the next driver.

- -DACK0-DACK3, -DACK5-DACK7

  A DMA Acknowledge (-DACK) is driven active by the system DMA controller in response to a DMA request (DRQ). For a DMA device the '-DACK' acts as a DMA device select, and the system DMA controller drives addresses and control signals on the I/O channel. For alternate controllers the '-DACK' indicates that the alternate controller has been granted use of the I/O channel following proper activation of the '-master' control line.

  Note: A DMA device should not honor its DACK, if its DRQ is disabled.

- DRQ8; -DACK8

  See "I/O Slot Uniqueness" on page 6-54 and "Coprocessor Arbitration" on page 6-56.

- -MASTER

  This signal is used by an alternate controller to show that the system board must tri-state its address and command lines so that the alternate controller may drive the I/O channel after receiving a '-DACK' from the system DMA controller. After receiving the '-DACK', a alternate controller pulls the '-master' line low. After waiting 60 nanoseconds, it may drive the address and data lines, and 125 nanoseconds after addresses are valid it may issue a Read or Write command. When the alternate controller has completed its cycle, it must tri-state all address, data, and data transfer control lines before turning off '-master' (high).

- T/C (Terminal Count)

  This line provides a pulse on the I/O channel during a Read or Write command. It shows that the terminal count of the current DMA operation has been reached. This represents the last byte to be transferred of a pre-programmed DMA block transfer. A DMA adapter must have a valid 'DACK' before 'T/C' is valid.

- IRQ3-IRQ7, IRQ9-IRQ12, IRQ14, IRQ15

  The interrupt request lines signal the processor that an I/O device requires attention. They occur on priority levels. For more details see "Interrupt Priority" on page 5-57.

- -I/O CH CK

  -I/O Channel Check generates a nonmaskable interrupt (NMI) to the system processor if an uncorrectable system error is indicated. '-I/O CH CK' is driven low momentarily, and it must be active for at least 2 I/O channel clock (CLK) periods to be detected. '-I/O CH CK' is valid on the transition high to low. Every device that uses '-I/O CH CK' must have a programmable status latch, and a programmable enable or disable latch.

  Note: Channel-attached RAM *does not* save status, and it holds '-I/O CH CK' active until it is cleared by the NMI handler software.

- CLK

  The RT PC system board uses 4.77 MHz with a 33% duty cycle.

  Note: I/O channel timing parameters will be specified independent of the I/O channel clock frequency to allow I/O device compatibility in different systems. I/O channel clocks may vary across PC family system boards and will be between 4-8MHz.

- -REFRESH

  This line is driven low on the system board to indicate a refresh cycle to I/O channel-attached RAM. See "Refresh Operation for Channel-Attached RAM" on page 6-26 for details of refresh operations. (See "I/O Slot Uniqueness" on page 6-54 for Coprocessor slot).

- RESET DRV

  Reset Driver is used to reset all system devices to an initial state. At power-on time 'RESET DRV' goes high and stays active for 100-500 ms after all voltage lines are within operating limits. When 'RESET DRV' is active, I/O adapters must turn off or tri-state all output drivers.

  Note: Software initiated reset commands to specific I/O channel adapters or options may be less than 100 milliseconds, if the adapter has a lower reset requirement.

- OSC

  The Oscillator is a high speed clock with a 14.31818 Mhz frequency and a 50% duty cycle.

- POWER

The following voltages are available on the system board I/O channel:

+5 Vdc ± 5%, located on 5 connector pins

-5 Vdc ± 10%, located on 1 connector pin

+12 Vdc ± 5%, located on 1 connector pin

-12 Vdc ± 10%, located on 1 connector pin

GND (Ground), located on 6 connector pins

I/O Channel Data Transfer

Data can be transferred on the I/O Channel either directly by the system processor, or by using Direct Memory Access.

System Processor

The system processor is on the processor board and attaches to the processor channel. The Input/Output channel controller interfaces the processor channel to the I/O channel and to the I/O subsystem. The system processor owns the I/O channel and will transfer control to the system arbiter when a valid request is active.

Input/Output Channel Controller (IOCC)

The IOCC transfers commands, addresses and data between the processor channel, the I/O channel and the I/O subsystems. It provides a centralized arbiter function for the various I/O subsystem and I/O channel users. It also contains the system arbiter, the system DMA controller and the interrupt controller.

Direct Memory Access (DMA)

DMA is a method of transferring large blocks of sequential data between two devices with minimal interference to the system processor. One participant in the transfer is usually memory. The other is an adapter or option which supplies or receives the data. DMA capability permits simultaneous use of I/O adapters and the system processor.

System DMA Controller

The system DMA controller supports both DMA devices and alternate controllers. The system DMA controller resides on the system board and is considered part of the IOCC. Two 8237 DMA controllers are used for this function. See the system board section for more detailed information. The system DMA controller must request use of the I/O channel from the system arbiter.

DMA Controller

A DMA controller is either the system DMA controller or an alternate controller.

DMA Types

Two types of DMA are used in this system:

DMA Device — A DMA device is an adapter or option *without* the ability to handle addressing or I/O controls. It uses the system DMA controller for addressing and for I/O channel control. The DMA device drives the data on the channel when directed by the system DMA controller.

Alternate Controller — An alternate controller is an adapter or option *with* the ability to handle its own data addressing and I/O channel control. All alternate controllers must be 16-bit devices (attached to SD0-SD15), drive '-SBHE' and honor 'I/O CH RDY'.

Note: A DMA adapter is either a DMA device or an alternate controller.

Alternate Controller Operation

When there is a need to transfer data via DMA using an alternate controller adapter, the program (software) for the adapter starts a preprogrammed block transfer. The software obtains the starting address in memory, the block transfer length and the direction of the transfer. It then loads all the proper parameters into the adapters DMA controller function and into the proper IOCC registers. Now the software instructs the hardware to take over. The hardware activates its 'DRQ' line enough times to transfer the complete block of data.

When the 'DRQ' line is activated, the system DMA controller receives the request, sets its priority along with any other active requests and acts in turn on the requests. When the alternate controller receives its '-DACK', it will activate '-Master'. The IOCC now tri-states all I/O channel address lines, data lines and proper control signals. Then the alternate controller logic drives the I/O channel address and control signals and drives or receives data as specified. After each byte of data is transferred, the alternate controller tri-states all its drivers, removes -Master and returns control of the I/O channel to the system DMA controller.

The hardware repeats this sequence of events until the complete preprogrammed block of data is transferred. On the last DMA cycle 'T/C' is active to indicate to the IOCC and to the system processor that the DMA block transfer is complete.

An alternate controller that desires to handle 8-bit and 16-bit devices must honor '-MEMCS16 and -I/OCS16'. The alternate controller determines how and when the data should be multiplexed and

provides the controls to multiplex data to D7-D0 for data transfers between 8-bit and 16-bit devices. This is only required if the adapter is used across the 16-bit PC family line.

Notes:

1. See "Arbitration" on page 5-33 for details of the system arbiter function.
2. Alternate controllers usually have an I/O device mode, therefore they can be controlled by the system processor or another alternate controller.
3. See required "DMA Request Clock Synchronization Logic" on page 6-64.

RT PC Coprocessor

Channel-attached processors such as the 286 Coprocessor are generally viewed as alternate controllers which have the additional attributes of executing code, servicing interrupts, and controlling other channel-attached devices. Remember, the system processor is on the processor board.

Note: Alternate controllers may also have a device mode. When in device mode, the alternate controller acts like any other I/O device.

Device

A device is any channel-attached option or adapter that responds to the address and control of the system processor or a DMA controller. Each device is assigned a unique address range to which it may respond. Once a device recognizes its own unique address on the channel, it is said to be selected, and it is then required to obey the data transfer controls. A device will store data during a Write operation and provide data during a Read operation. Three types of devices are generally referenced:

- DMA devices
- Memory devices
- I/O devices

DMA Device

A DMA device is a device that has the ability to request a DMA operation (+ DRQ). It requests use of the I/O channel when it needs to transfer a block of data with another device (usually memory). DMA block transfers are achieved in the same manner as alternate controller operations (see "Alternate Controller Operation" on page 6-16) except that the system DMA controller provides addressing and data transfer control signals for the transfer. The DMA device must be ready to provide or receive the data before it requests the use of the I/O channel. It can not drive the 'I/O CH RDY' line not ready for additional setup time.

Memory Device

A memory device is a RAM, ROM or EPROM. A memory device can drive the 'I/O CH RDY' line not ready to extend the time to provide or receive data.

I/O Device

An I/O device is an I/O adapter or option which is able to provide or receive data under the control of the system processor, system DMA controller or an alternate controller. An I/O device can drive the 'I/O CH RDY' line not ready to extend the time for providing or receiving data.

I/O Channel Cycle Default Timings

During a data transfer cycle between the IOCC and a memory or I/O device, if the device does not drive 'I/O CH RDY' not ready, then the IOCC must provide the following default cycle timings:

		Min	Typ.	Max
Case 1.	For 8-bit devices -IOR, -MEMR pulse width	540ns	565ns	1000ns
Case 2.	For 8-bit devices -IOW, -MEMW pulse width	540ns	565ns	1000ns
Case 3.	For 16-bit memory devices -MEMR pulse width	235ns	250ns	600ns
Case 4.	For 16-bit I/O devices -IOR pulse width	175ns	190ns	600ns
Case 5.	For 16-bit memory devices -MEMW pulse width	235ns	250ns	600ns
Case 6.	For 16-bit I/O devices -IOW pulse width	175ns	190ns	600ns

- *Any Alternate Controller* which transfers data with a 16-bit I/O space device *must* receive the signal '-IOCS16' and provide the "short" default cycle timing, Case 4 and Case 6.
- *Any Alternate Controller* which transfers data with a 16-bit memory space device *must* receive the signal '-MEMCS16' and provide the "short" default cycle timing, Case 3 and Case 5.
- *Any Device* that cannot store data on a write cycle within the minimum cycle times described above (Case 2, 5, and 6) *must* use I/O CH RDY "not ready" to stretch the cycle.
- *8-bit Memory Devices must* provide valid data within 250ns of an active read command or else drive I/O CH RDY "not ready" to stretch the cycle.
- *8-bit I/O Devices must* provide valid data within 425ns of an active read command or else drive 'I/O CH RDY' not ready to stretch the cycle.
- *16-bit Memory Devices must* provide valid data within 200ns of an active read command or else drive 'I/O CH RDY' not ready to stretch the cycle.
- *16-bit I/O Device must* provide valid data within 135ns of an active read command or else drive 'I/O CH RDY' not ready to stretch the cycle.
- *The system DMA controller must* support memory default timing.
  - Assume memory READ/WRITE access = max. (250ns)
  - Assume DMA controller READ = max. (250ns) data valid
  - Assume DMA controller WRITE = max. (200ns) data setup.
- *Any memory which supports DMA must* be able to read or write data within 250ns of an active command or else drive 'I/O CH RDY' not ready to stretch the cycle.
- *Any DMA device must* be able to read data within 250ns of an active command and it will have a minimum of 200ns data setup before the end of an '-IOW' command. Data must be sampled on the trailing edge of '-IOW' to guarantee valid data setup.

I/O Channel Protocols

This section describes the protocols that channel participants use to carry out channel operations. Channel operations included are:

- DMA read/write
- Memory read/write
- I/O read/write
- Interrupt
- Refresh operations
- Error detection
- Reset.

Figure 6-4 on page 6-21 illustrates the different channel participants, their relationship to the channel structure, important functional controls and protocols.

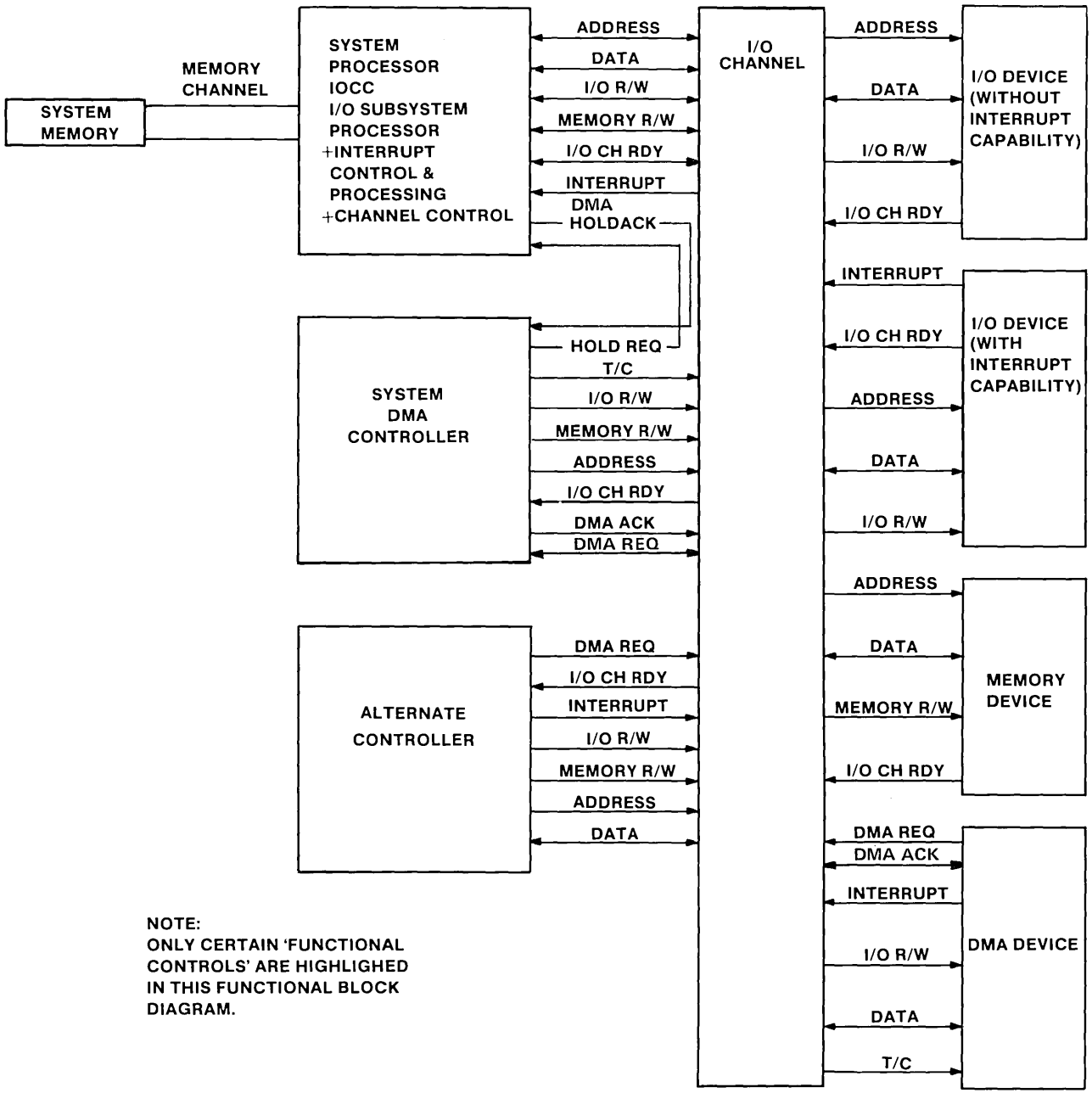


Figure 6-4. Protocol Functional Block Diagram

Memory Read Operation

For a Memory Read Operation, the DMA controller must drive the address lines and insure they are stable on the channel before the Memory Read command is activated. When the Memory Read command is detected by the memory device, it drives its data onto the I/O channel. If a memory device is unable to drive data onto the I/O channel within the time specified for a normal operation, the 'I/O CH RDY' line must be driven not ready by the memory device. This will extend the memory read operation and allow additional time before the memory device must drive data onto the I/O channel. The data must be valid on the channel before the memory device releases the 'I/O CH RDY' line to a ready state. The memory device must hold the data lines active until the Memory Read command goes inactive.

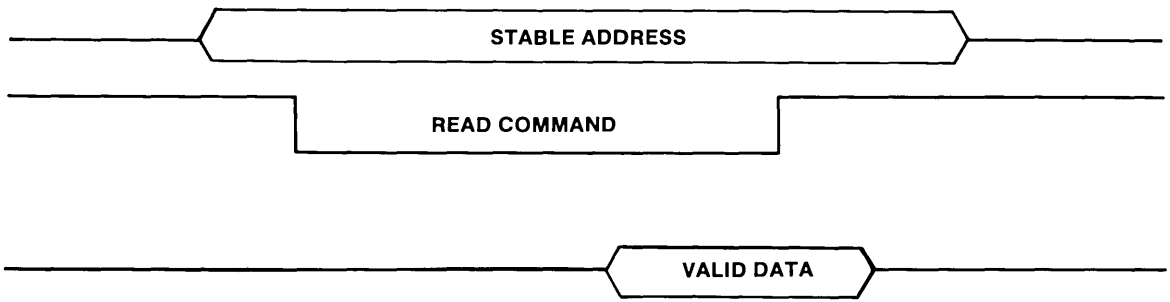


Figure 6-5. Memory Read Operation Sequence

Memory Write Operation

For a Memory Write Operation, the DMA controller must drive both the address and the data lines active. During a memory write operation, the Memory Write command is driven active after the address lines are stable, but before the data lines are valid on the channel. It is preferable to drive the data active before the Memory Write command goes active.

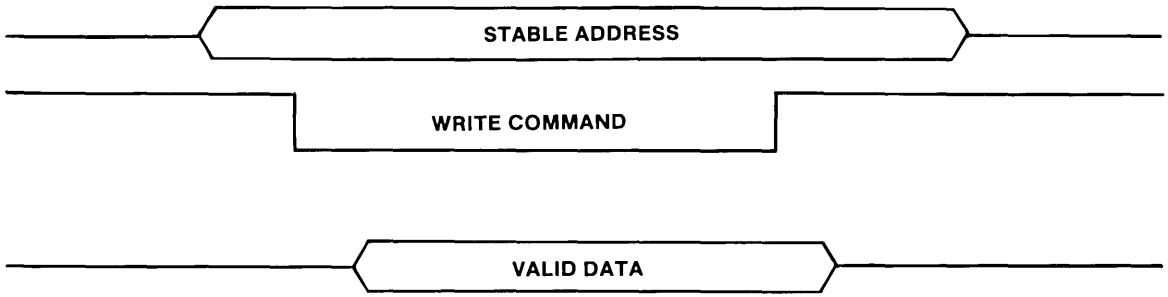


Figure 6-6. Memory Write Operation Sequence

I/O Read Operation

For a Read Operation, the DMA controller must drive the address lines and insure they are stable on the channel before the command (I/O or memory) is activated. When the Read Command is detected by the device, it drives its data onto the I/O channel. If a device is unable to drive data onto the I/O channel within the default cycle time, the 'I/O CH RDY' line must be driven not ready by the device. This will extend the I/O read operation and allow additional time before the device must drive data onto the I/O channel. The data must be valid on the channel before the device releases the 'I/O CH RDY' line to a ready state. The device must hold the data lines active until the Read Command goes inactive.

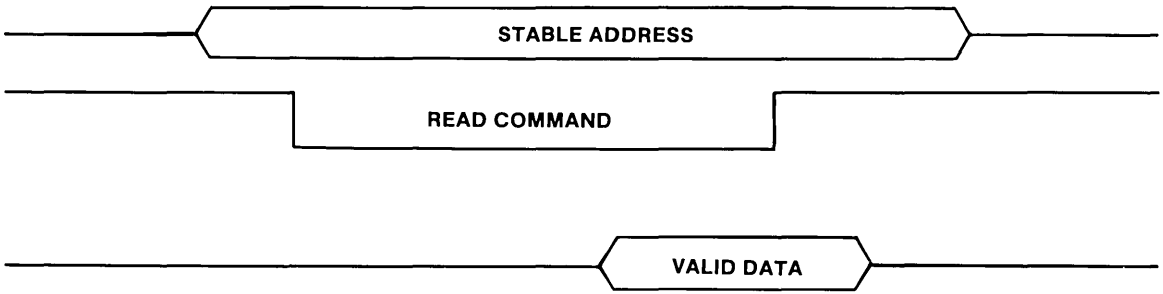


Figure 6-7. I/O Read Operation Sequence

I/O Write Operation

For a Write Operation, the DMA controller must drive both the address and the data lines active. During a write operation, the Write Command (I/O or memory) is driven active after the address lines are stable, but before the data lines are valid on the channel. A device must sample the data lines at the trailing edge of the Write Command to ensure data will be valid for all circumstances. If a device is unable to store the data within the default cycle time, the 'I/O CH RDY' line must be driven not ready by the device. This will extend the I/O write operation and allow the device additional time to store the data. The DMA controller must maintain both address and data lines active and stable for the duration of the Write Command.

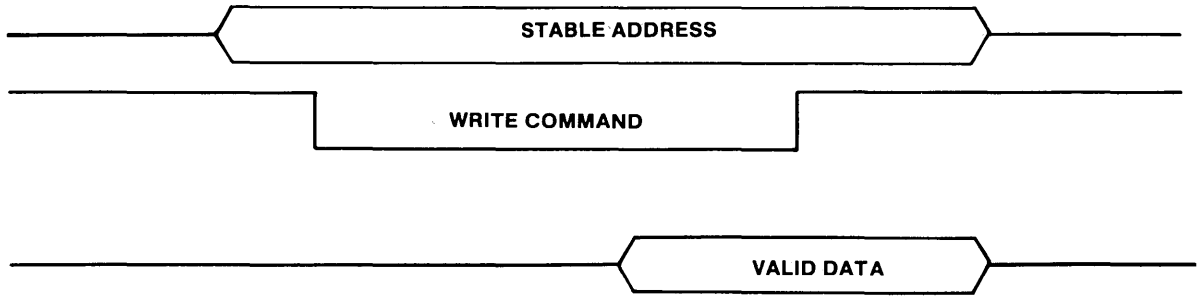


Figure 6-8. I/O Write Operation Sequence

Refresh Operation for Channel-Attached RAM

Channel refresh operations occur at the minimum rate of 256 refresh cycles every 4ms. The 256 refresh cycles consists of all combinations of row addresses SA0-SA7. This will support refresh for all RAM cards using standard 16K x 1, 64K x 1, and 256K x 1 RAM modules.

'-REFRESH' is driven low by the system board to indicate that channel-attached RAM must start a refresh operation. The '-REFRESH' line is used to enable a RAS (Row Address Strobe) to all memory banks when '-MEMR' goes active, and also to disable all CAS (Column Address Strobe) controls.

The refresh operations are normally done in distributed bursts of five refresh cycles. However, if the refresh controller is delayed in getting control of the I/O channel, the maximum refresh burst is 17 refresh cycles. Refresh can be held off by 'I/O CH RDY', but system performance may suffer.

See Figure 6-9 on page 6-27 for Refresh timing.

Error Detection

When active, the 'I/O CH CK' line raises a non-maskable interrupt to the processor. This is the highest priority interrupt and takes precedence over other system operations.

Reset

The 'RESET DRV' control line forces all system devices to an initial state. At power on time, 'RESET DRV' is active high for at least 100 msec after all voltages are within operating limits. The states of the channel signals at the end of a reset are:

Signal	State
ADDRESS BUS	Unknown State
DATA BUS	Tri-state
IRQ(n)	Active (High)
DRQ (n)	Inactive (Low)
DACK(n)	Inactive (High)
BALE	Always High
AEN	Inactive (Low)
RD/WR	Inactive (High)
CLK	4.77 MHz (free running)
I/O CH CK	Inactive (High)
Other Control Signals	Inactive

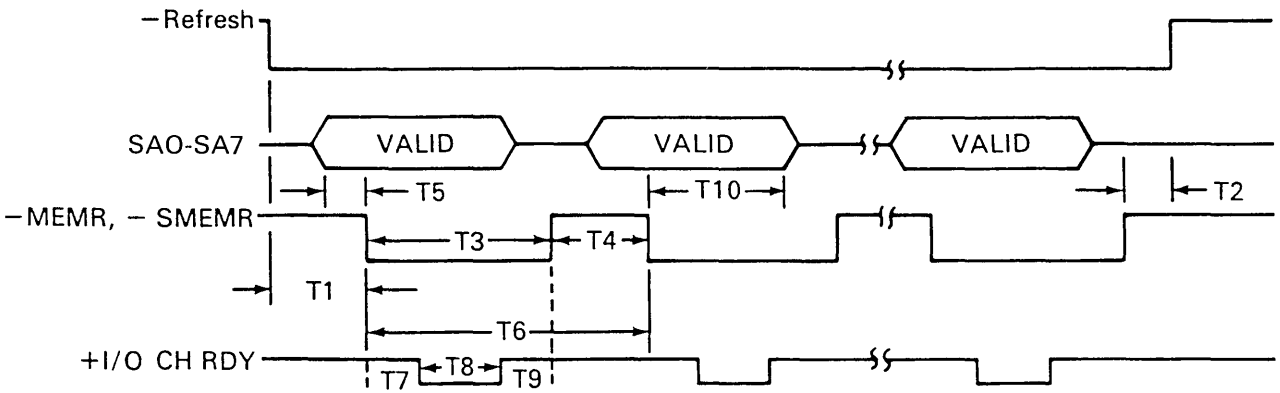


Figure 6-9. Refresh Timing

Note: Only the system board logic can initiate a refresh timing cycle.

Symbol	Parameter Description	Min.	Max.
T1	Refresh set-up to command	200	—
T2	Refresh hold from command	50	—
T3	Refresh read low	400	—
T4	Refresh read high	200	—
T5	Refresh address set-up	125	—
T6	Refresh cycle time	600	—
T7	I/O CH RDY not ready from command	0	60
T8	I/O CH RDY pulse width	0	600
T9	Read command off from I/O CH RDY	0	—
T10	Address valid after command	300	—

Note: All times are specified in nanoseconds.

Timing Diagrams

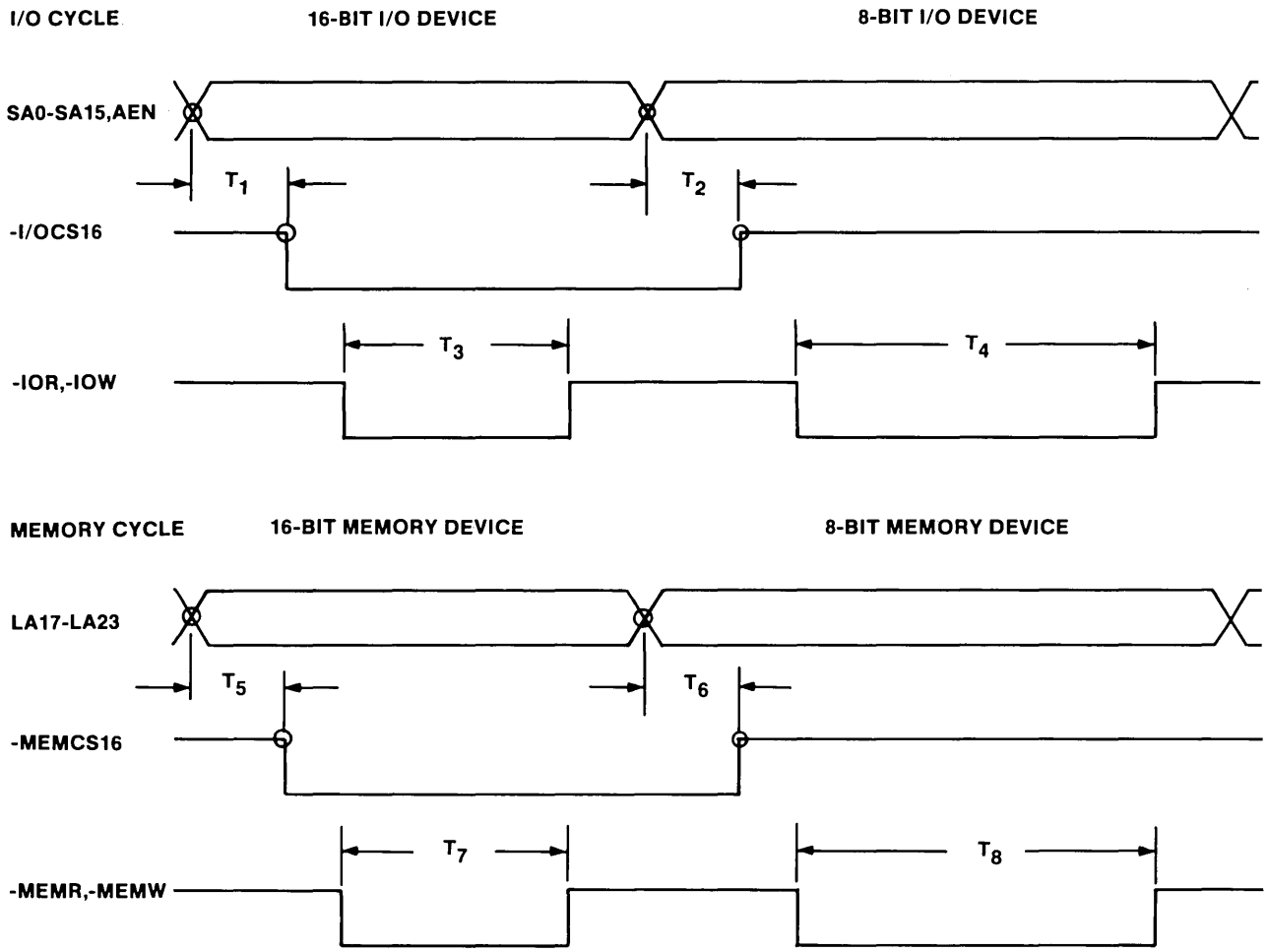


Figure 6-10. I/O And Memory Default Cycle Timing

Symbol	Parameter Description	Min.	Typ.	Max.
T1	-I/OCS16 active from valid address	0	—	90
T2	-I/OCS16 inactive from address change	0	—	90
T3	Fast I/O cycle command pulse width (default)	175	350	600
T4	Slow I/O cycle command pulse width (default)	540	630	1000
T5	-MEMCS16 active from valid address	0	—	90
T6	-MEMCS16 inactive from address change	0	—	90
T7	Fast memory cycle command pulse width (default)	235	350	600
T8	Slow memory cycle command pulse width (default)	540	630	1000

Note: All times are specified in nanoseconds.

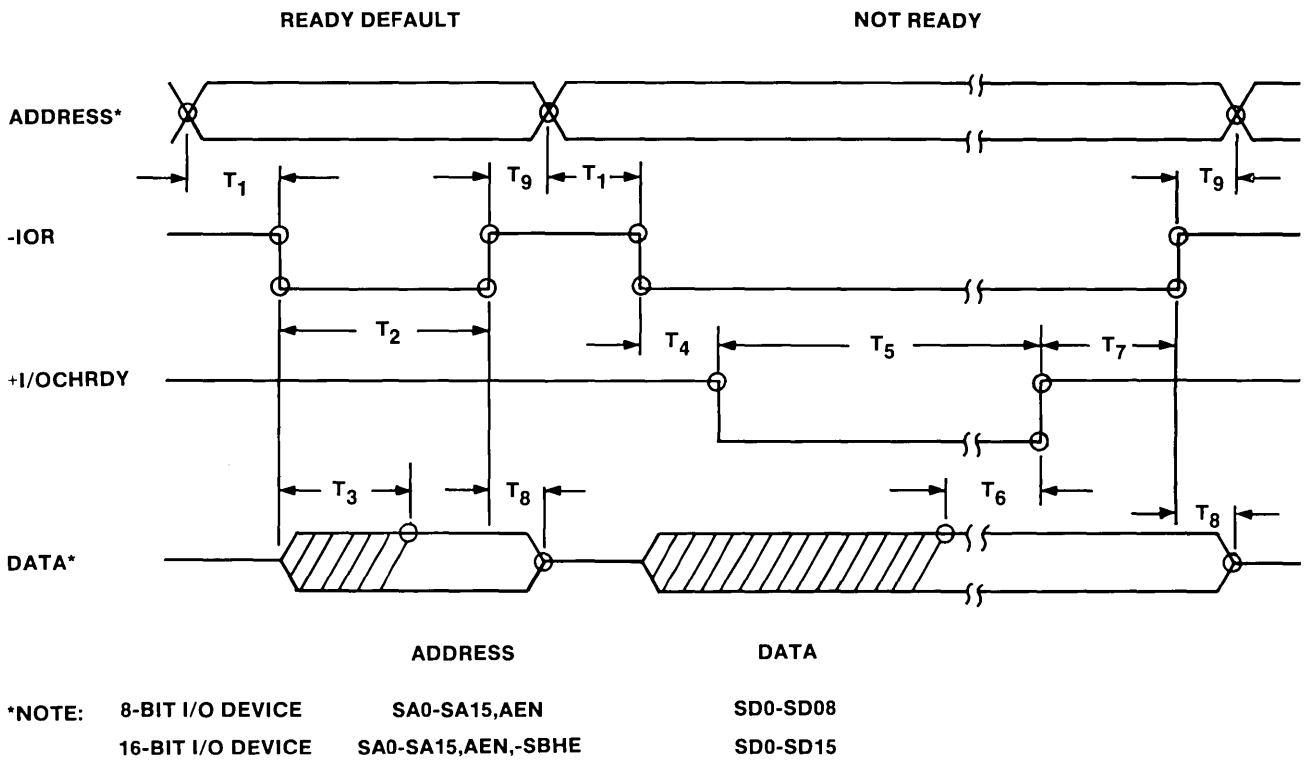


Figure 6-11. I/O Read Operation

Symbol	Parameter Description	Min.	Typ.	Max.
T1	Address valid before -IOR is active	90	140	
T2	-IOR pulse width default: 8-bit device	540	630	1000
	16-bit device	175	350	600
T3	Data valid from -IOR default: 8-bit device	0		425
	16-bit device	0		135
T4	I/O CH RDY not ready from -IOR: 8-bit device	0		150
	16-bit device	0		60
T5	I/O CH RDY not ready pulse width	0		2100
T6	Data valid before I/O CH RDY ready	0	50	
T7	-IOR inactive from I/O CH RDY ready	0	140	
T8	Data tri-state from -IOR inactive	0		45
T9	Address hold time from -IOR inactive	40	60	

Note: All times are specified in nanoseconds.

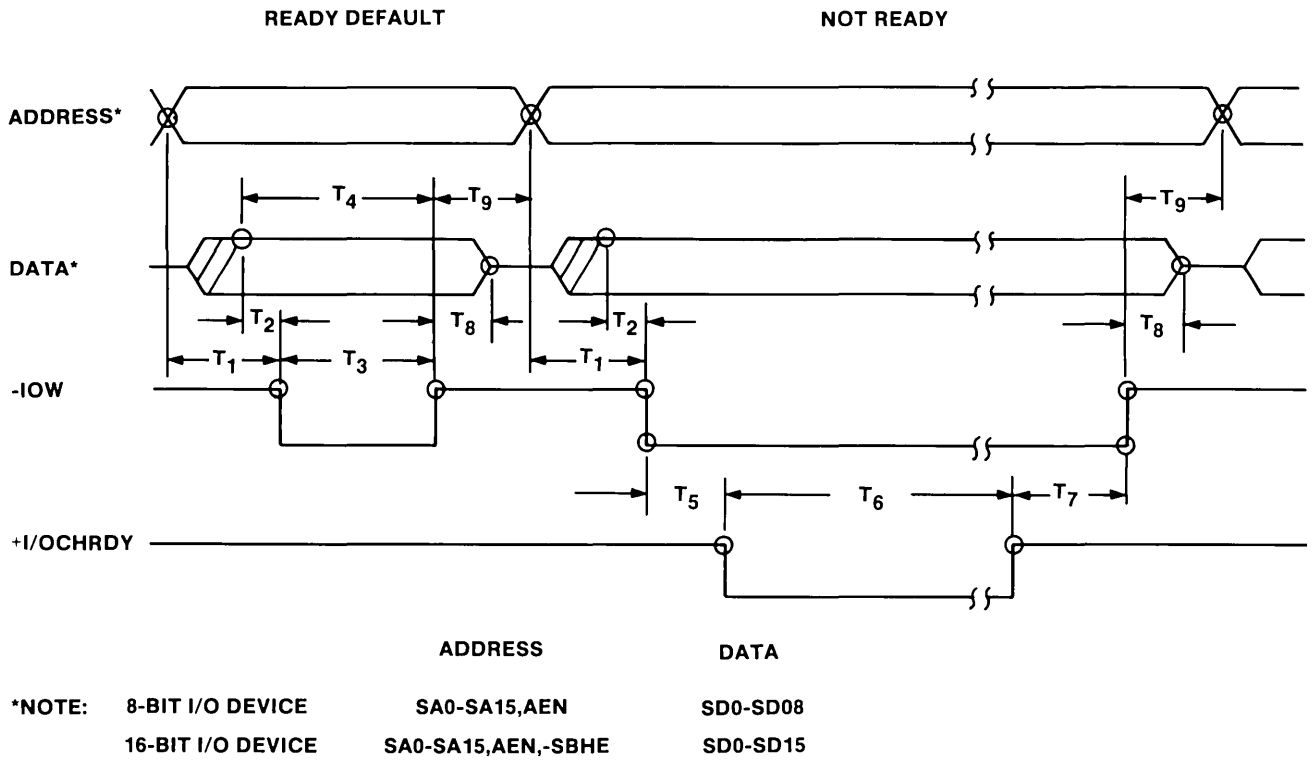


Figure 6-12. I/O Write Operation

Symbol	Parameter Description	Min.	Typ.	Max.
T1	Address valid before -IOW is active	90	140	
T2	Data valid before -IOW is active	0	40	
T3	-IOW pulse width default: 8-bit device	540	630	1000
	16-bit device	175	350	600
T4	Data setup time before -IOW inactive: 8-bit device	540	670	
	16-bit device	175	390	
T5	I/O CH RDY not ready from -IOW: 8-bit device	0		150
	16-bit device	0		60
T6	I/O CH RDY not ready pulse width	0		2100
T7	-IOW inactive from I/O CH RDY ready	0	140	
T8	Data hold time from -IOW inactive	40	60	
T9	Address hold time from -IOW inactive	40	60	

Note: All times are specified in nanoseconds.

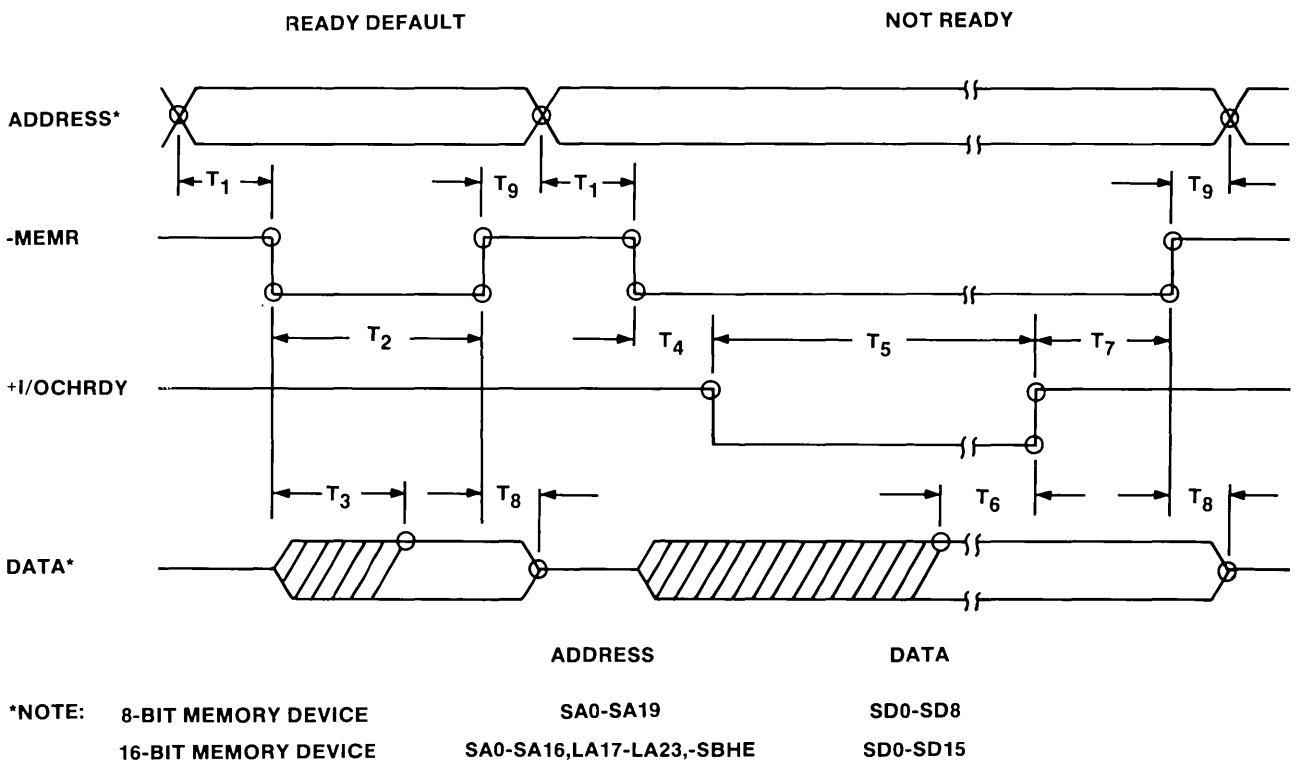


Figure 6-13. Memory Read Operation

Symbol	Parameter Description	Min.	Typ.	Max.
T1	Address valid before -MEMR is active	90	140	
T2	-MEMR pulse width default: 8-bit device	540	630	1000
	16-bit device	235	350	600
T3	Data valid from -MEMR default: 8-bit device	0		250
	16-bit device	0		200
T4	I/O CH RDY not ready from -MEMR: 8-bit device	0		150
	16-bit device	0		60
T5	I/O CH RDY not ready pulse width	0		2100
T6	Data valid before I/O CH RDY	0	50	
T7	-MEMR inactive from I/O CH RDY ready	0	140	
T8	Data tri-state from -MEMR inactive	0		45
T9	Address hold time from -MEMR inactive	40	60	

Note: All times are specified in nanoseconds.

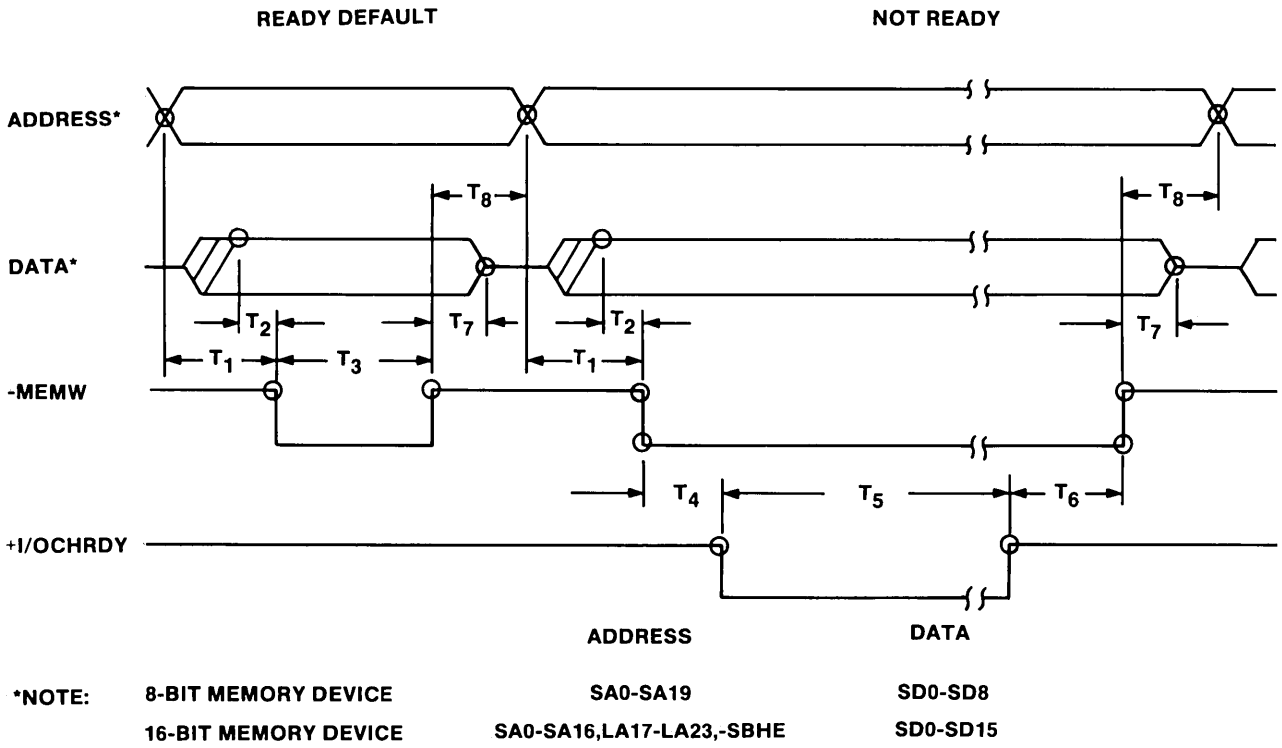


Figure 6-14. Memory Write Operation

Symbol	Parameter Description	Min.	Typ.	Max.
T1	Address valid before -MEMW is active	90	140	
T2	Data valid before -MEMW is active	0	40	
T3	-MEMW pulse width default: 8-bit device	540	630	1000
	16-bit device	235	350	600
T4	I/O CH RDY not ready from -MEMW: 8-bit device	0		150
	16-bit device	0		60
T5	I/O CH RDY not ready pulse width	0		2100
T6	-MEMW inactive from I/O CH RDY ready	0	140	
T7	Data hold time from -MEMW inactive	40	60	
T8	Address hold time from -MEMW inactive	40	60	

Note: All times are specified in nanoseconds.

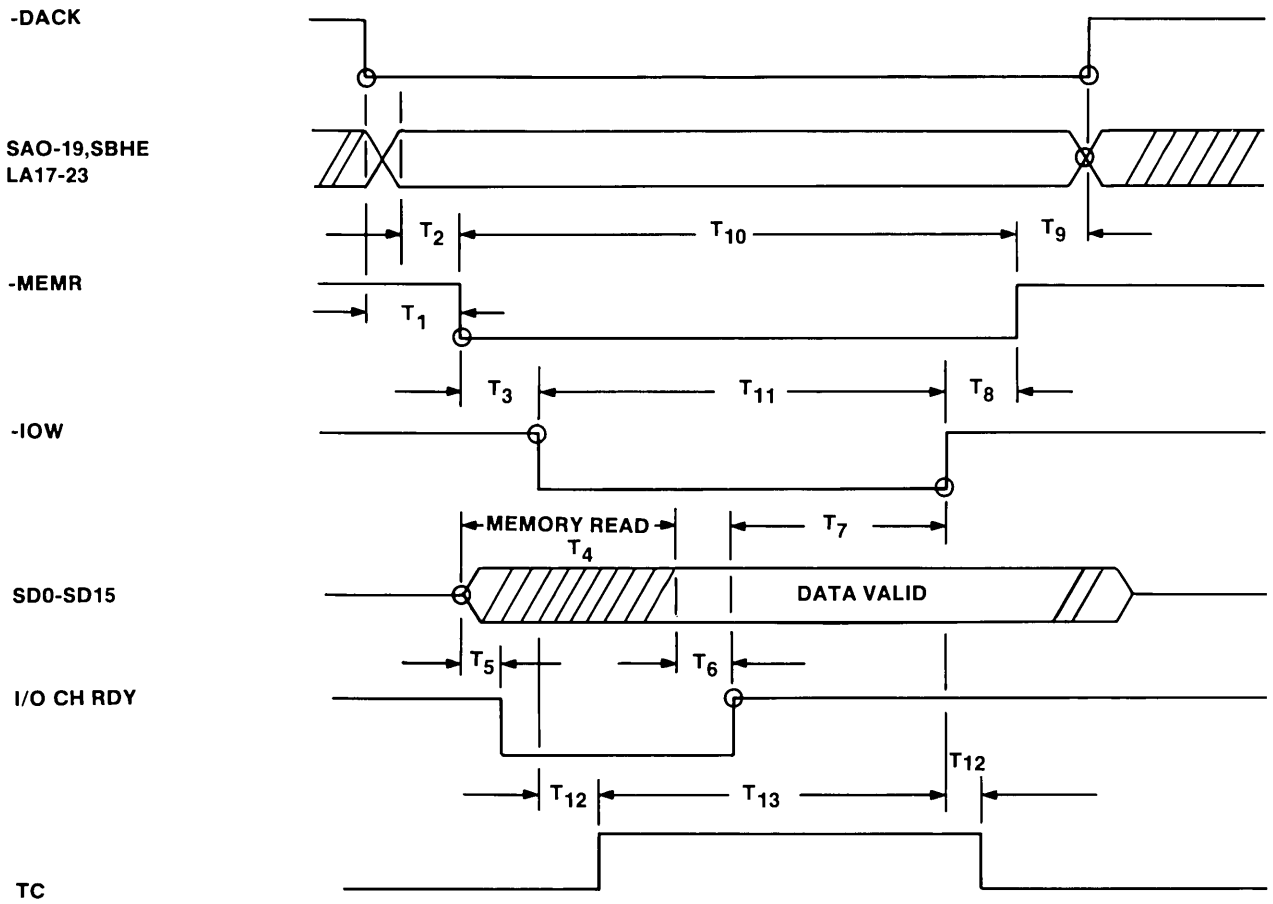


Figure 6-15. DMA Read Operations

Symbol	Parameter Description	Min.	Max.
T1	-DACK active before -MEMR is active	50	–
T2	Address -SBHE or AEN valid before commands	90	–
T3	-MEMR active before -IOW active	0	–
T4	Memory access time (default)	0	250
T5	I/O CH RDY not ready from -MEMR active	0	60
T6	I/O CH RDY ready from data valid	0	–
T7A	Valid data setup before -IOW inactive (ready case)	175	–
T7B	Valid data setup before -IOW inactive (default case)	200	–
T8	-MEMR inactive after -IOW inactive	25	–
T9	Address -SBHE/AEN/DACK hold time after -MEMR inactive	40	–
T10	-MEMR minimum pulse width (see note 2)	500	–
T11	-IOW minimum pulse width	260	–
T12	T/C valid	−50	50
T13	T/C minimum pulse width	250	

Notes:

1. All times are specified in nanoseconds.
2. For DMA accesses to system memory, -MEMR signal will not go active on the I/O channel.

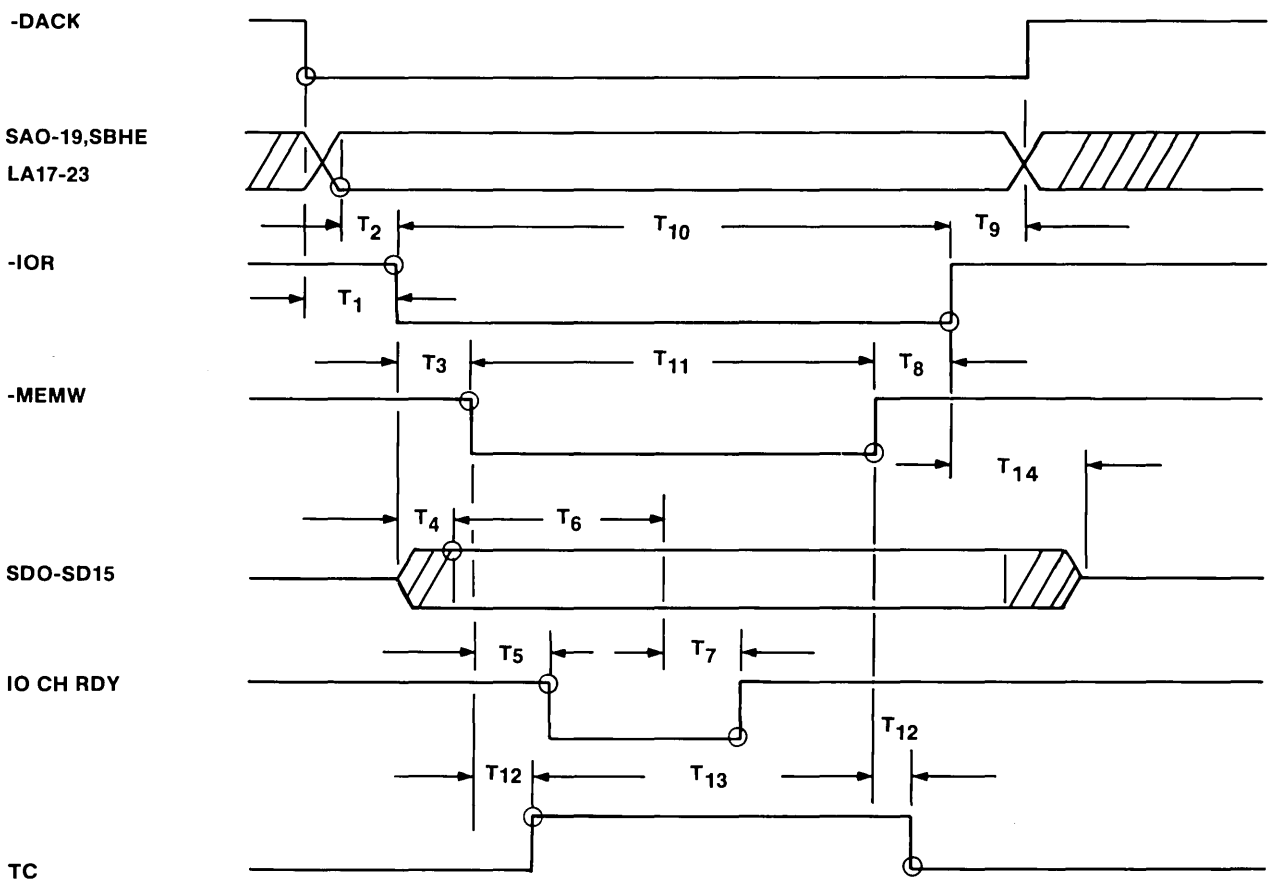


Figure 6-16. DMA Write Operations

Symbol	Parameter Description	Min.	Max.
T1	-DACK active before -IOR is active	50	–
T2	Address -SBHE or AEN valid before commands	90	–
T3	-IOR active before -MEMW active	200	–
T4	Data valid from -IOR	0	200
T5	I/O CH RDY not ready from -MEMW active	0	60
T6	Time for memory to store data (default)	0	250
T7	I/O CH RDY ready from memory data stored	0	–
T8	-IOR inactive after -MEMW inactive	25	–
T9	Address, -SBHE, AEN, DACK hold time after -IOR inactive	40	–
T10	-IOR minimum pulse width	500	–
T11	-MEMW minimum pulse width (see note 2)	260	–
T12	T/C valid	–50	50
T13	T/C minimum pulse width	250	
T14	Data bus tri-state after -IOR inactive	0	45

Notes:

1. All times are specified in nanoseconds.
2. For DMA accesses to system memory, -MEMW signal will not go active on the I/O channel.

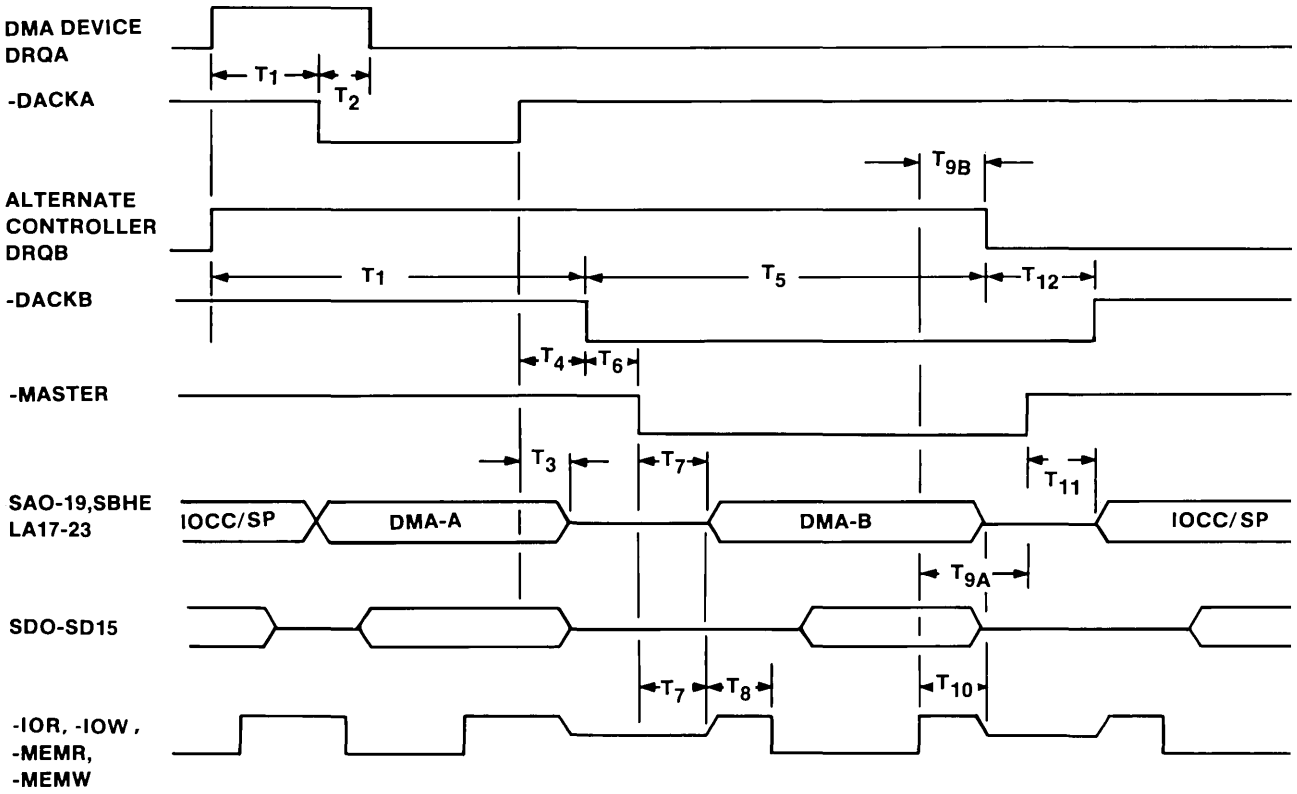


Figure 6-17. Bus Arbitration

Symbol	Parameter Description	Min.	Max.
T1	DMA acknowledge from DRQ active (see note 1)	–	–
T2	DRQ inactive from -DACK active (see note 6)	0	125
T3	Bus and commands tri-state from -DACK off (see note 2)	–	–
T4	-DACK inactive to next DACK active	0	–
T5	DRQ inactive from -DACK active (see notes 3 and 4)	–	–
T6	-DACK active to -Master active	0	–
T7	-Master active before MasterB drives bus	60	–
T8	Address valid before Read/Write command active	125	–
T9A,B	-Master and DRQB inactive from Read/Write command inactive (see note 3)	–	–
T10	Bus and commands tri-state before -Master inactive	0	–
T11	-Master inactive to IOCC drives bus (see note 4)	0	–
T12	DRQ (alt controller) inactive to -DACK inactive (see note 5)	–	–

Note: All times are specified in nanoseconds.

Notes:

1. When multiple requests are active, the DACK response is a function of the IOCC and the system configuration. This means several higher priority devices may get acknowledged first.
2. If the IOCC is switching from one IOCC cycle to another (system processor or DMA device), it will not need to tri-state the channel. If it is transferring control to an alternate controller it must tri-state the channel before issuing the next DACK.
3. A alternate controller must hold DRQ and -Master active when it needs to control the channel. This includes meeting address and data hold-time requirements.
4. The IOCC may drive the channel immediately, go idle, or issue another DACK as soon as possible after -Master and DRQ are off, and the IOCC has turned off the original DACK (DACKB in the example).
5. The IOCC deactivates -DACK as soon as possible after the DRQ has been deactivated. The IOCC rearbitrates when DRQ turns off.

6. This specification is for single cycle DMA transfers. DRQ for multiple cycle DMA transfers may turn off as late as 60ns. after the last command goes active.
7. SP denotes system processor.

Interrupts

This section describes interrupts and their usage in RT PC. There are descriptions of the interrupt sharing architecture and circuit operation.

I/O Channel Interrupt Operations

The interrupt request (IRQ) lines signal the system processor or a Coprocessor that an I/O adapter requires service. When an adapter requests service, the interrupt controller interrupts the system processor. Then the system processor acknowledges the interrupt. It is then serviced by the appropriate interrupt routines.

All I/O adapters must have a programmable and readable status latch (bit) which shows the interrupt status. The adapter must also have a programmable and readable latch (bit) which can enable or disable the interrupt capability of the adapter. System reset on the 'Reset Dvr' line must reset the status latch, the enable latch and the interrupt line driver on all adapters.

RT PC supports two types of interrupts, shared and nonshared interrupts on all its levels. Only one I/O adapter with a nonsharing interrupt can have its interrupt enabled at the same time (per level). I/O adapters with shared IRQ's may have their IRQ's enabled simultaneously and operate concurrently. Mixing types of interrupts on the same level is allowed, if one type is disabled, when the other is enabled. RT PC has implemented shared interrupts to solve the problems with non-shared interrupts.

Nonshared Interrupts

With nonshared interrupts only one adapter can use an IRQ line. The normal state of an enabled nonshared IRQ is low. An interrupt request is generated by raising the IRQ line (from low to high) and holding it high until it is acknowledged by the processor. Then the interrupt service routine must clear the IRQ line (from high to low).

Note: Disabling a nonshared interrupt forces the IRQ line to go from low to high or tri-state level. The system interrupt controller assumes a valid interrupt has occurred. Therefore, an interrupt level must be masked off before an adapter interrupt is disabled, and the false interrupt condition must be cleared before normal processing can continue on that level.

Shared Interrupts

Shared interrupts allow a system to have multiple adapters using a common interrupt request line. To request service, a device sends a negative pulse (125 to 1000 nanoseconds) to the IOCC interrupt controller. The interrupt controller recognizes the interrupt on the low-to-high transition of the pulse. An adapter must set its interrupt status latch bit before starting an interrupt or use it to start the interrupt. The adapter's interrupt status latch is reset and the interrupt level is reenabled by the interrupt service routine. The IRQ lines are driven with an open collector driver (or tri-state equivalent) and each I/O adapter must provide a 2.0K or 2.2K ohm +/- 10% pull up resistor on the driver output.

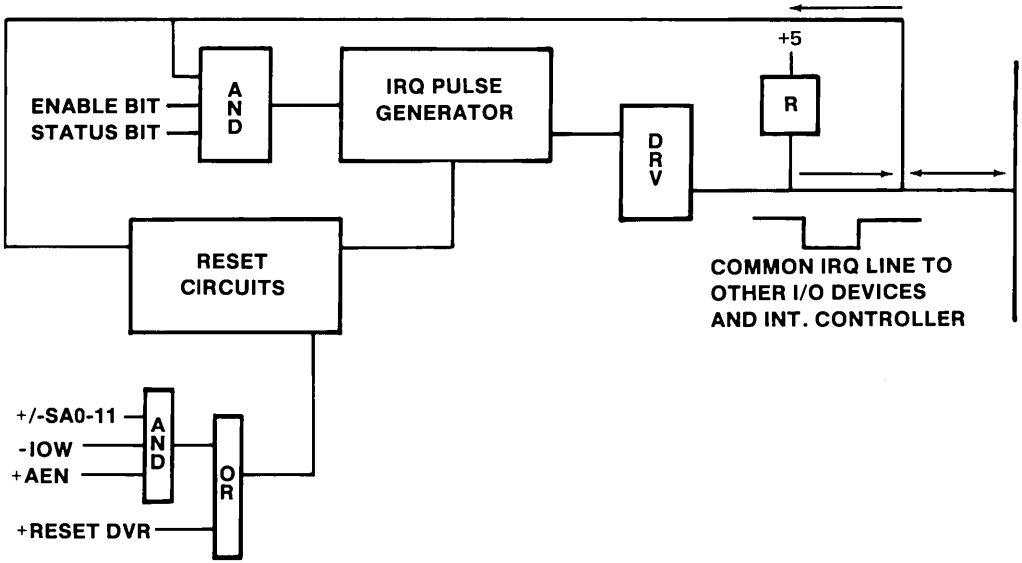


Figure 6-18. Block Diagram For IRQ Line

Figure 6-18 shows, conceptually, a method of controlling a device's interrupt request output to allow sharing of an interrupt request line. An interrupt originating from the adapter (status bit) is 'ANDed' with the adapter's interrupt enable bit, and the result is used to activate the interrupt request pulse generating circuits unless an interrupt is already active. The adapter requesting service must disable all adapters on the same interrupt level until the requesting adapter is finished. If another adapter has an interrupt pending, it is sent immediately after the trailing edge of the I/O Write which reenables that interrupt level. The I/O Write which is issued by the software interrupt level service routine reenables all devices on that interrupt level. Each interrupt level has a common I/O address that reenables all adapters with one I/O Write instruction.

The following is a list of the I/O addresses for the interrupt levels.

Interrupt Level	Hex Address
3	2F3
4	2F4
5	2F5
6	2F6
7	2F7
9	2F2
10	6F2
11	6F3
12	6F4
14	6F6
15	6F7

Bits SA0 through SA10 are the only bits required to be decoded for this address. They must be 'ANDed' with '-IOW' and ' + AEN'. When the I/O address is written, it is called Interrupt Level Enable (ILE). The ILE output must be 'ORed' with ' + Reset Dvr' as part of the reset circuits. When the ILE is written, no data is required.

When a shared interrupt adapter is enabled, the proper ILE must be issued before the device can begin interrupt operations.

I/O Channel Address Maps and Assignments

Memory Address Map

Hex Address	Name	Function
000000 to 07FFFF	512K-bytes	I/O channel memory - IBM Personal Computer AT 512KB Memory Expansion Option
080000 to 09FFFF	128K-bytes	Additional I/O channel memory (Each Baseband Adapter uses 32K-bytes of this memory)
0A0000 to 0BFFFF	128K-bytes video RAM	Reserved for graphics display buffer (Megapel display uses 0B8000 through 0BFFFF)
0C0000 to 0DFFFF	128K-bytes I/O expansion ROM	Reserved for ROM on I/O adapters. PC Network Adapter uses 0CC000 - 0CDFFF (primary) and 0DC000 - 0DDFFF (alternate). 3278/79 Emulation Adapter uses 0CE000 - 0CFFFF, 0D0000 - 0D1FFFF, 0D2000 - 0D3FFF, and 0D4000 - 0D5FFF.
0E0000 to 0EFFFF	64K-bytes Reserved on I/O channel memory	Duplicated code assignment at address FE0000 for Coprocessor option.
0F0000 to 0FFFFF	64K-byte ROM on the I/O channel memory	Duplicated code assignment at address FF0000 for Coprocessor option.
100000 to BFFFFF	Maximum memory up to 11M-bytes	I/O channel memory reserved for Coprocessor using Expansion Option.
C00000 to C3FFFF		Megapel adapter data.

Figure 6-19 (Part 1 of 2). Memory Address Map

Hex Address	Name	Function
D00000 to D1FFFF		Advanced Monochrome Graphics Display
D20000 to D3FFFF		Advanced Color Graphics Display
D40000 to D5FFFF		Coprocessor video relocation buffer
D60000 to D7FFFF		Megapel adapter program
D80000 to D9FFFF		Extended Monochrome Graphics Display Adapter
E00000 to E1FFFF		System expansion ROM
E40000 to E4FFFF		Multiprotocol Adapter #1
E50000 to E5FFFF		Multiprotocol Adapter #2
E80000 to E9FFFF		5080 Attachment Adapter
EA0000 to EBFFFF		S/370 Host Interface Adapter
FE0000 to FFFFFF		TCW pass through (page mode)

Figure 6-19. Memory Address Map

Note: All I/O channel memory addresses C00000-FFFFFF are reserved for RT PC system use.

I/O Address Map

Hex Address	Device
0000-003F	Reserved
0040-004F	Coprocessor Access
0440-044F	Coprocessor Access
0840-084F	Coprocessor Access
0C40-0C4F	Coprocessor Access
0050-00FF	Reserved
0140-014F	Token Ring Adapter #2
0150-015F	Advanced Color Graphics Display
0160-016F	Advanced Monochrome Graphics Display
0170-0177	Fixed-disk adapter #2, ESDI, Extended ESDI
01C0-01CF	Token Ring Adapter #1
01E8-01EF	Streaming Tape Drive Adapter
01F0-01F7	Fixed-disk adapter #1, ESDI, Extended ESDI
0278-027F	Parallel printer port 2
02D0-02DF	3278/79 Emulation Adapter
02F0-02F7	Interrupt sharing
02F8-02FF	Serial port 2
0300-031F	Prototype card
0360-0367	PC Network #1
0368-036F	PC Network #2

Figure 6-20 (Part 1 of 3). I/O Address Map

Hex Address	Device
0370-0377	Diskette controller secondary
0378-037F	Parallel printer port 1
03B0-03BF	Monochrome Display and Printer Adapter
03B0-03DF	Enhanced Color Graphics Adapter
03F0-03F7	Diskette controller primary
03F8-03FF	Serial port 1
0510-052F	Multiprotocol Adapter #1
0550-0557	286 Coprocessor communication with system processor
0570-0577	Extended ESDI secondary
05F0-05F7	Extended ESDI primary
06D0-06DF	3278/79 Emulation Adapter (alternate)
06F0-06F7	Interrupt sharing
0910-092F	Multiprotocol Adapter #2
0930-094F	Megapel display
0950-095F	SCSI primary
0AD0-0ADF	3278/79 Emulation Adapter (alternate)
0D10-0D2F	Extended Monochrome Graphics Display Adapter
0D50-0D5F	SCSI secondary
0E20-0E2F	S/370 Host Interface Adapter
0E90-0E9F	5080 Attachment Adapter
0ED0-0EDF	3278/79 Emulation Adapter (alternate)
11C0-11CF	Token Ring Adapter #3

Figure 6-20 (Part 2 of 3). I/O Address Map

Hex Address	Device
11D0-11DF	Token Ring Adapter #4
1230-124F	First address range (multiport async)
2230-224F	Second address range (multiport async)
3230-324F	Third address range (multiport async)
4230-424F	Fourth address range (multiport async)

Figure 6-20 (Part 3 of 3). I/O Address Map

Note: The I/O address of the Enhanced Color Graphics Adapter overlaps those of the Monochrome Adapter when it operates in PC monochrome compatibility mode.

DMA Channels

The following is a list of the DMA channels and the devices associated with those channels.

Channel	Device
0	Serial Port A, SCSI, Extended ESDI
1	Serial Port B, Multiprotocol Adapter, SCSI, Extended ESDI
2	Diskette Drive, Serial Port A
3	PC Network, Serial Port B, SCSI, Token Ring #4, Extended ESDI
5	Mutliprotocol Adapter, 5080 Adapter, S/370 Adapter, Token Ring #1, SCSI, Megapel
6	5080 Adapter, S/370 Adapter, Token Ring #2, SCSI, Megapel
7	Extended Monochrome Graphics Display Adapter, 5080 Adapter, S/370 Adapter, SCSI, Megapel, Token Ring #3
8	Coprocessor

Figure 6-21. DMA Channels

Interrupt Levels

The following is a list of the interrupt levels and what devices interrupt on the various levels.

Interrupt Level	Device
3	Serial Port 2, PC Network, Baseband Adapter
4	Serial Port 1, Baseband Adapter
5	Parallel Port 2, Baseband Adapter
6	Diskette Drive, Baseband Adapter
7	Parallel Port 1, Monochrome/Printer, Baseband Adapter
9	PC Network, (Multiport Async), Enhanced Color Graphics Adapter, 3278/79 Emulation Adapter, Baseband Adapter, (5080 Peripheral Adapter)
10	(Multiport Async), (Multiprotocol Adapter), (5080 Adapter), (S/370 Adapter), (5080 Peripheral Adapter)
11	(Advanced Monochrome Graphics Display), (Multiport Async), (Advanced Color Graphics Display), (Extended Monochrome Graphics Display Adapter), (Multiprotocol Adapter), (Megapel), (SCSI), (Token Ring), (5080 Peripheral Adapter)
12	(Streaming Tape Drive Adapter), (Token Ring Adapter), (SCSI), (Extended ESDI Fixed Disk)
14	Fixed Disk, ESDI Fixed Disk, (Extended ESDI Fixed Disk)
15	(Coprocessor)

Figure 6-22. Interrupt Levels

Note: The parentheses indicate a shared interrupt device.

I/O Slot Uniqueness

One slot (Coprocessor) on each system board has some unique assignments that could make it incompatible with some PC cards. The slot has one new DMA channel (8), and it will not support memory board refresh. The board preassignment and unique pin assignment for the I/O slots is given below.

Slot	Unique	Connectors	Assignment
1	No	62-Pin/40-Pin	Disk/Diskette Adapter
2	No	62-Pin/40-Pin	Option
3	No	62-Pin	Option
4	No	62-Pin/40-Pin	Option
5	No	62-Pin/40-Pin	Option
6	No	62-Pin	Option
7	No	62-Pin/40-Pin	Option
8	Yes	62-Pin/40-Pin	Coprocessor/Option

Figure 6-23. IBM 6150 System Boards

Slot	Unique	Connectors	Assignment
1	No	62-Pin	Option
2	No	62-Pin/40-Pin	Option
3	No	62-Pin/40-Pin	Option
4	No	62-Pin/40-Pin	Option
5	Yes	62-Pin/40-Pin	Coprocessor/Option
6	No	62-Pin/40-Pin	Disk/Diskette Adapter

Figure 6-24. IBM 6151 System Board

Pin	I/O Slot Standard	I/O Slot Coprocessor
B19	REFRESH	+ SPK
D14	DACK 7	DRV
D15	DRQ 7	- DACK 8
		+ DRQ 8

Figure 6-25. Unique Pin Assignments

Coprocessor Arbitration

The Coprocessor has the lowest DMA request priority on the I/O channel. However, when it is executing out of system memory or I/O channel attached RAM, it can hold on to the channel for multiple cycles to improve performance.

When the Coprocessor requires an I/O channel cycle, it drives the 'DRQ8' signal active. When the channel becomes idle (that is, no other DRQ, PIO (system processor) request or refresh request is pending) the system arbiter drives the '-DACK8' signal active. The Coprocessor drives the '-master' signal active and then follows the ground rules for alternate controllers when driving the channel. If the channel remains idle, the Coprocessor executes until it is finished using the channel. It will then turn off the 'DRQ8' signal and end the alternate controller operation normally.

If, however, the system arbiter receives a new request while the Coprocessor is active, it turns off the '-DACK' signal. The Coprocessor may complete any I/O channel operation that is in progress, and then it must tri-state its channel drivers and turn the'DRQ8' and '-master' signals off.

When the 'DRQ8' signal turns off, the system arbiter will acknowledge another requestor. If the request was from the system processor, the IOCC may drive the channel immediately .

The Coprocessor must wait at least 250 nanoseconds before re-issuing a DMA request.

Connector/Pin Description and Electrical Characteristics

Refer to the following tables (starting in Figure 6-26) for the list of I/O signals and the associated electrical characteristics.

Note: Capacitive load should not exceed 20 picofarads (pf) per signal per adapter. In an 8 adapter configuration channel capacitance will be 200pf maximum allowing 40pf on the system board.

Signal Name	I/O Pin	Driver (Iol MIN ma)	Driver Type	Receiver (Iil MAX ma)
SA0	A31	24	TS	-0.4
SA1	A30	24	TS	-0.4
SA2	A29	24	TS	-0.4
SA3	A28	24	TS	-0.4
SA4	A27	24	TS	-0.4
SA5	A26	24	TS	-0.4
SA6	A25	24	TS	-0.4
SA7	A24	24	TS	-0.4
SA8	A23	24	TS	-0.4
SA9	A22	24	TS	-0.4
SA10	A21	24	TS	-0.4
SA11	A20	24	TS	-0.4
SA12	A19	24	TS	-0.4
SA13	A18	24	TS	-0.4
SA14	A17	24	TS	-0.4
SA15	A16	24	TS	-0.4
SA16	A15	24	TS	-0.4
SA17	A14	24	TS	-0.4
SA18	A13	24	TS	-0.4
SA19	A12	24	TS	-0.4
AEN	A11	24	TS	-0.4
I/O CH RDY	A10	24	OC	-2.0
SD0	A09	24	TS	-0.4
SD1	A08	24	TS	-0.4
SD2	A07	24	TS	-0.4
SD3	A06	24	TS	-0.4
SD4	A05	24	TS	-0.4
SD5	A04	24	TS	-0.4
SD6	A03	24	TS	-0.4
SD7	A02	24	TS	-0.4
-I/O CH CK	A01	24	OC	-2.0

Figure 6-26. 62-Pin Connector Side A

TS Tri-state drivers will be used for the class of signals which have multiple sources that are enabled one at a time.

TSP	Tri-state drivers for command lines will have pullups on the system board.
OCI	Open collector drivers used for IRQ sharing require a 2.0K or 2.2K ohm ± 10% pullup on the I/O adapter. There is a 5K pullup on the system board.
X	Optional.
OC	Open collector drivers are required for signals that may be dotted together and simultaneously driven by multiple sources. The system board must supply a pull-up resistor for these signals.

Signal Name	I/O Pin	Driver (Iol MIN ma)	Driver Type	Receiver (Iil MAX ma)
GND	B31	—	—	—
OSC	B30	X	X	-2.0
+5	B29	—	—	—
BALE	B28	24	OC	-0.4
T/C	B27	8	X	-0.4
-DACK2	B26	8	X	-0.4
IRQ3	B25	24	OCI	-0.8
IRQ4	B24	24	OCI	-0.8
IRQ5	B23	24	OCI	-0.8
IRQ6	B22	24	OCI	-0.8
IRQ7	B21	24	OCI	-0.8
CLK	B20	24	X	-2.0
-REFRESH	B19	40	OC	-2.0
DRQ1	B18	24	TS	-0.2
-DACK1	B17	8	X	-0.4
DRQ3	B16	24	TS	-0.2
-DACK3	B15	8	X	-0.4
-IOR	B14	24	TSP	-0.4
-IOW	B13	24	TSP	-0.4
-SMEMR	B12	24	TSP	-0.4
-SMEWM	B11	24	TSP	-0.4
GND	B10	—	—	—
+12	B09	—	—	—
(RESERVED)	B08	—	—	—
-12	B07	—	—	—
DRQ2	B06	24	TS	-0.2
-5	B05	—	—	—
IRQ9	B04	24	OCI	-0.8
+5	B03	—	—	—
RESET DRV	B02	24	X	-2.0
GND	B01	—	—	—

Figure 6-27. 62-Pin Connector Side B

Signal Name	I/O Pin	Driver (Iol MIN ma)	Driver Type	Receiver (Iil MAX ma)
Signal	C20	—	—	—
+ 5 (see note)	C19	—	—	—
+ 5 (see note)	C18	24	TS	-0.4
SD15	C17	24	TS	-0.4
SD14	C16	24	TS	-0.4
SD13	C15	24	TS	-0.4
SD12	C14	24	TS	-0.4
SD11	C13	24	TS	-0.4
SD10	C12	24	TS	-0.4
SD9	C11	24	TS	-0.4
SD8	C10	24	TSP	-0.4
-MEMW	C09	24	TSP	-0.4
-MEMR	C08	24	TS	-0.4
LA17	C07	24	TS	-0.4
LA18	C06	24	TS	-0.4
LA19	C05	24	TS	-0.4
LA20	C04	24	TS	-0.4
LA21	C03	24	TS	-0.4
LA22	C02	24	TS	-0.4
LA23	C01	24	TS	-0.4
-SBHE				

Figure 6-28. 40-Pin Connector Side C

Note: Pins C19 and C20 are additional + 5 volt power pins provided on RT PC.

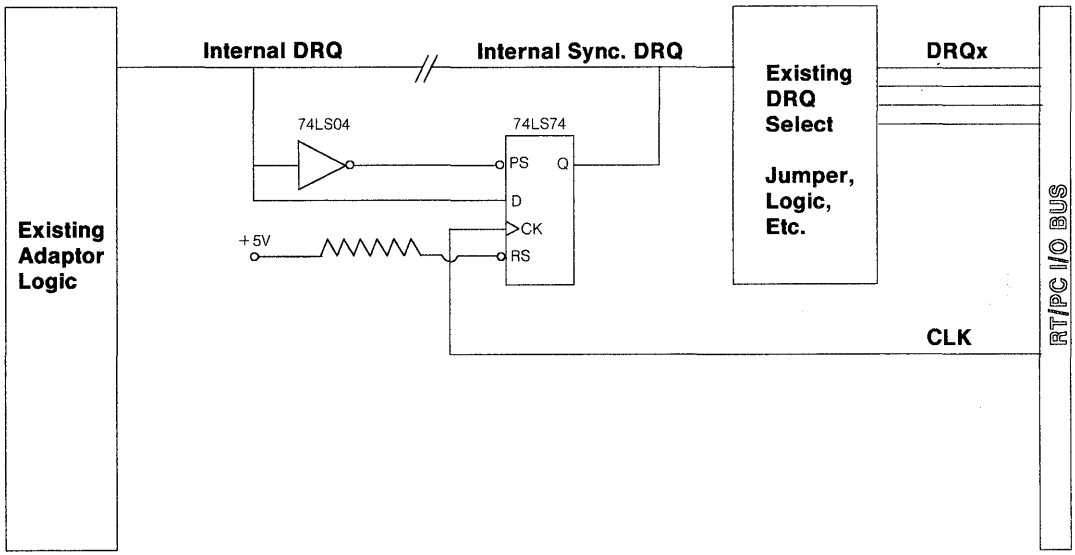
Signal Name	I/O Pin	Driver (Iol MIN ma)	Driver Type	Receiver (Iil MAX ma)
GND (see	D20	—	—	—
note)	D19	—	—	—
GND (see	D18	—	—	—
note)	D17	24	OC	-2.0
GND	D16	—	—	—
-MASTER	D15	24	TS	-0.2
+5	D14	8	X	-0.4
DRQ7	D13	24	TS	-0.2
-DACK7	D12	8	X	-0.4
DRQ6	D11	24	TS	-0.2
-DACK6	D10	8	X	-0.4
DRQ5	D09	24	TS	-0.2
-DACK5	D08	8	X	-0.4
DRQ0	D07	24	OCI	-0.8
-DACK0	D06	24	OCI	-0.8
IRQ14	D05	24	OCI	-0.8
IRQ15	D04	24	OCI	-0.8
IRQ12	D03	24	OCI	-0.8
IRQ11	D02	24	OC	-2.0
IRQ10	D01	24	OC	-2.0
-I/OCS16				
-MEMCS16				

Figure 6-29. 40-Pin Connector Side D

Note: Pins C19 and C20 are additional ground pins provided on RT PC.

DMA Request Clock Synchronization Logic

The following logic or its equivalent is required by all DMA adapters implementing the alternate controller function.



Section 7. System IPL ROM

CONTENTS

About this Section

This section describes the IPL ROM programming necessary to start the RT PC from a powered down condition. It contains the layout for the power on self test (POST) control block and lists the IPL ROM display codes.

System IPL ROM

The RT PC Initial Program Load (IPL) Read Only Memory (ROM) contains the programming necessary to start the RT PC from a powered down condition. It also provides other functions in support of diagnostics and user convenience re-IPL operations. The functions performed by the IPL ROM and the system requirements it imposes are also discussed.

POST Control Block

Internal ROM operation and results are recorded in the POST control block. The POST control block is located in Random Access Memory (RAM) and is created during the execution of ROM power-on self tests. The layout and system memory address assignments of the POST control blocks are shown in Figure 7-1 on page 7-5. At the end of ROM execution, the POST control block contains the following:

- A four word entry for each power-on self test, as shown in Figure 7-2 on page 7-6.
  1. An ID that identifies which power-on self test was executed.
  2. An error code which indicates whether or not an error occurred during power-on self tests and, if so, what that error was.
  3. Two configuration words which specify the address, interrupt levels and Direct Memory Access (DMA) levels used on the board.
- R/W memory size
- List of bad blocks on 128K-byte boundaries
- Machine type (6150, 6151)
- Attached keyboard ID number, if available from keyboard
- The IPL device number used for IPL
- Soft IPL flag byte.

The loadable POSTs execution adds an entry in the same format for each loadable POST executed.

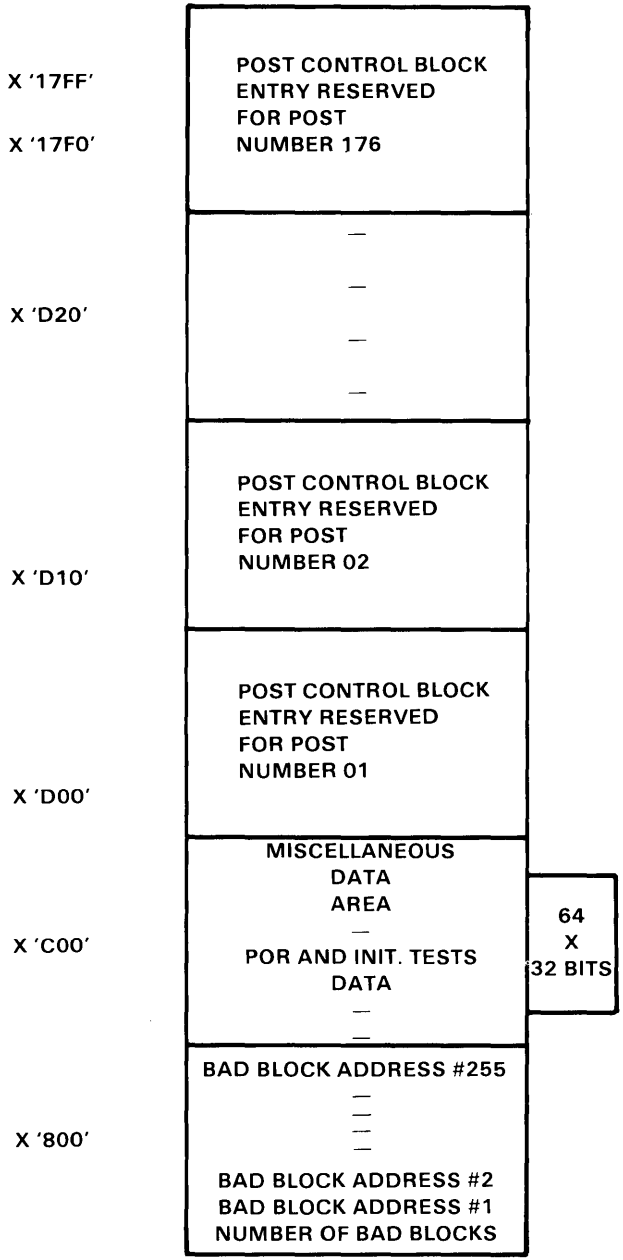


Figure 7-1. Layout of POST Control Block

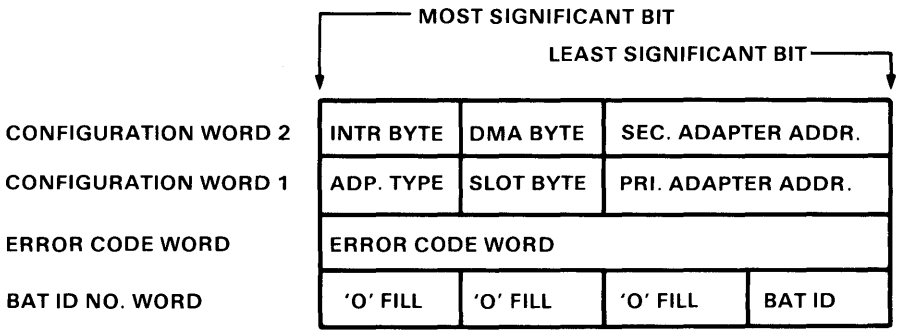


Figure 7-2. Layout of the POST Control Block Section Dedicated to One Post

POST Control Block Fields

The fields defined in Figure 7-2 have the following contents.

Slot Byte

The slot byte contains the slot number where the primary and secondary adapters were found. The least significant 4 bits are the primary adapter slot number and the most significant 4 bits are the secondary adapter slot number. A value of X'F' indicates that no adapter was found.

Interrupt Byte

The interrupt byte contains the interrupt level used by the primary and secondary adapters. The least significant 4 bits are the interrupt level of the primary adapter and the most significant 4 bits are the interrupt level of the secondary adapter. For two levels used on a single adapter, both levels are indicated but refer to a single adapter. A value of X'8' indicates that no interrupts are used on the adapter or that the adapter is not present.

DMA Byte

The DMA byte contains the DMA levels used by the primary and secondary adapters. The least significant 4 bits are the DMA level of the primary adapter and the most significant 4 bits are the DMA level of the secondary adapter. A value of hex "F" indicates that no DMA level is used.

Primary Adapter Address

This 2 byte field contains the I/O address of the primary adapter. Hex 'FFFF' indicates no adapter responds to this address.

Secondary Adapter Address

This 2 byte field contains the I/O address of the secondary adapter. Hex 'FFFF' indicates no adapter responds to this address.

Adapter Type

This field contains the adapter type number.

The 4 port asynchronous adapter overflows these definitions. The configuration data on this adapter conforms to this format for the primary and secondary adapters. Equivalent information on the third and fourth adapters is provided in a second information block in the POST control block as though these were a separate adapter type.

Error Code Word

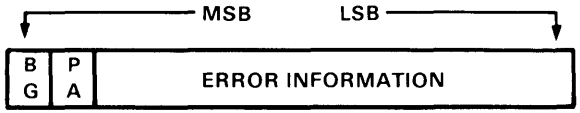


Figure 7-3. Error Code Word

The error code word is defined as follows:

Bit 0 Bad/Good Bit

1 = Bad
0 = Good

Bit 1 Present/Absent Bit

1 = Present
0 = Absent

Bits 2-31 Thirty bits of error information

NVRAM Used by ROM

The portions of the real time clock Non Volatile Random Access Memory (NVRAM) referenced by ROM are:

F000880E - F000883D CRC checked and compared to F000883E - F000883F
F0008828 - F000882C IPL source table

The IPL ROM code looks at the 5 entries in the IPL source table in NVRAM (H'F0008828' to 'F000882C') after the CRC check bytes are verified as correct for NVRAM contents. Each byte in the source table is compared to a ROM list of eligible candidates for IPL and when a match is found then IPL is attempted using that device. If IPL is not completed from that device then the next entry in the source table is checked for IPL device use. This continues until all entries in the table have been compared. Then ROM uses a ROM default table of IPL devices, which is also used when the NVRAM CRC is incorrect. The NVRAM IPL source table is used only once per IPL or re-IPL and is evident by the LED indicators showing a 21. The byte values used in the NVRAM IPL source table are:

H 'F0'	First diskette drive on adapter 1.
H 'F1'	Second diskette drive on adapter 1.
H 'F2'	First diskette drive on adapter 2.
H 'F3'	Second diskette drive on adapter 2.
H 'D0'	First disk drive on adapter 1.
H 'D1'	Second disk drive on adapter 1.
H 'D2'	First disk drive on adapter 2.
H 'D3'	Second disk drive on adapter 2.

Indicator Operation and Codes

The codes displayed on the POST indicators during IPL ROM execution are listed in "Display Codes" on page 7-32 and are further explained in the diagnostics package. All codes not specifically listed are reserved for RAS purposes and future loadable POST releases. Indicators use is reserved for diagnostic and error recovery. The indicators should be blank during normal system software operation.

System State

The machine will be left with:

- Interrupts disabled
- All good memory initialized with good ECC
- I/O subsystem I/O devices initialized and quiescent.

Use of ROM Code by Loadable Software

Parts of the ROM code may be usable by subsequent software under certain conditions. Since most of the ROM code was written in a high level language, certain restrictions apply to the use of the modules in the ROS entry point table. These restrictions and certain conventions required are detailed in "Restrictions and Conventions for Externally Callable Routines" on page 7-17. The user must provide the proper environment for any ROM modules used. A table of supported entry points (the ROS entry point table) is placed in ROM to support the system software IPL. This table is a list of entry points of modules which may be used. The table location is not guaranteed and must be located by the pointer stored at X'C18' in the POST control block. The order of the entry points in the POST control block is constant. As with all reused code, the ROM code may do a slightly different task than expected, so it should be thoroughly understood before use.

For all ROM modules called from loaded code, the following must be observed:

- ROM must be mapped into the address space.
- Access must be made in real address mode.
- Access must be made through the ROS entry point table pointer at X'C18' in the POST control block.
- Location X'C28' must either contain 0 or the user must provide appropriate run time routines for the ROM execution.

Disk and Diskette Sector Read Module Reuse

The ROS entry point table contains entry points for modules to read sectors from disk and diskette. These modules accept the following parameters:

- The object memory device
- The physical address on the memory device as required by the memory adapter
- A buffer pointer for the destination of the data obtained.

The modules do a read operation to the selected memory device with up to three retries. The module returns an error code to the calling program to indicate resolution. The modules will not do physical block number translation or any write operation.

System Expansion ROM Function

To provide functional growth for future releases, the ROM code provides a means of attaching additional ROM code on a I/O channel adapter. This feature enables IPL adapters other than Fixed Disk and Diskette Adapter to be used for IPL. Other uses are possible, but not encouraged. The I/O channel memory space address H'E00000' to H'E1FFFF' is reserved for expansion ROM. System ROM accesses this area on 8K-byte boundaries searching for a correct expansion ROM header pattern. When a correct pattern is found, the area is checked for a valid CRC and if valid, is treated as a ROM expansion area.

A valid ROM expansion area contains at least three major divisions within it. These are:

- Header

The header contains the recognition pattern, length, and CRC validation information without which the expansion code is not run.

- Power on expansion code

The power on expansion code is a section of code which is run as an extension to the in line IPL ROM operation. It has complete control of the machine when run. This code must be present, even if it is merely a return to the normal IPL ROM operation. There is a structured interface to return to IPL ROM, which is mandatory to continue operation.

- IPL expansion code

This code section can be called to access an IPL device, provided the correct setup for its use is done by the power on expansion code. The power on expansion code can store information in the POST control block to enable execution of the IPL expansion code in concert with the normal multiple IPL device selection provided for normal IPL devices.

Other code areas may exist in an expansion area for arbitrary uses, provided they do not interfere with the three functional areas mentioned above.

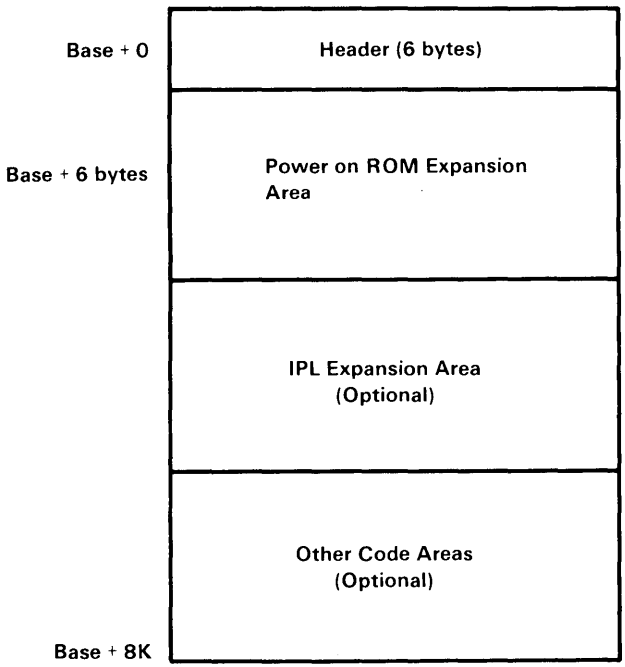


Figure 7-4. ROM Expansion View

Header Area

The header is a six byte long field which contains the recognition pattern, the length of the expansion ROM, and the CRC check value for the ROM. The recognition pattern consists of two bytes containing H'A5A5'. The CRC field is a two-byte field containing the CRC value for the expansion ROM size contained in the length field. The length field is a two-byte field containing the size of the expansion ROM in bytes. The six-byte header is not CRC checked by ROM, only the area after the header is checked for CRC. This area starts at BASE + 6 (see Figure 7-4) and includes all bytes as specified by the two-byte length in the header.

Power On Expansion Code

The power on expansion code area immediately follows the header area. It is of variable size and may occupy the remainder of its expansion ROM area. The entry point to this code must be the first byte of the area. The power on expansion code area may perform an IPL without returning to system IPL ROM or it may execute and return to IPL ROM execution. If no function is performed in this code area, it must contain at least a minimal set of code which returns to system ROM correctly. This area provides initialization and orderly linkage to existing system ROM for a future IPL adapter.

IPL Expansion Code

The IPL expansion code area is optional. If it is present, the power on expansion code execution must leave linkage information in the POST control block to allow IPL expansion code execution. The IPL expansion code area must contain the complete coding to cause an IPL or to return an architected return code if unable to IPL.

Other Expansion Code Areas

Any other code areas are optional. System ROM will not use the code but it may be used by other software. The expansion code area is not fixed at 8K-bytes and may be smaller or larger for module economy, up to 64K-bytes. It is checked at 8K-boundaries for existence and exists if the first bytes in an 8K area are H'A5A5'. Expansion code larger than 8K-bytes is valid if 'A5A5' does not occur on 8K boundaries. The recognition pattern is unlikely to occur in valid code, especially on 8k boundaries; the owner of such code must ensure that it does not if the expansion ROM is larger than 8K-bytes.

ROM Expansion Linkage Information

Once a ROM expansion area is recognized and verified by CRC, the entire area is loaded into system memory and ROM gives control to the first expansion code instruction.

The register convention is:

- R2 contains a pointer to the POST control block.
- R3 contains a pointer to the ROM entry table.
- R4 contains a pointer to the boot record area.
- R5 contains a pointer to the beginning of the ROM expansion area.
- R6 contains a pointer to the entry point of the ROM expansion area.

The expansion code having control of the machine at this point may:

- Execute instructions that set adapter devices as candidates for IPL, perhaps in NVRAM.
- Read a boot record from some device on its adapter into the boot record area (H'1800')
- Set the IPL device identification number in the PCB (H'C04')
- Copy the ROM entry table to memory and change the ROM disk or diskette module pointers in the expansion code area
- Change the ROM entry table pointer in the PCB (H'C18') so it points to the memory table. Set R2 and return to ROM IPL to continue.

When the expansion code sequence returns to ROM the register convention is:

- R2 is the return code.
  - Return code 0 signifies that the expansion code has completed its action and ROM should continue in the normal IPL manner.
  - Return code 1 signifies that the expansion code has completed its action. ROM should check the boot record area for a valid boot record plus use the IPL device ID in the Post control block as the usable device for loading executable code.
- R15 is the return to ROM address.

Note: ROM registers R6 through R15 must be saved after the expansion code receives control from ROM and registers R6 through R15 must be restored to the original values on return to the ROM.

ROM Location

The ROM address is set to real address X '800000' in its initial execution. It is possible to respecify the ROM address to another location. This may cause a system crash, since the code is compiled to run at X '800000'. Some of the code may operate transparently at another address, but this is not guaranteed since some of the code is assembler coded for a specific address. The memory management unit disallows any reference to ROM in translate mode.

The ROM entry point table provides for the use of ROM code by loaded code. It contains the following pointers, as 4 byte absolute addresses:

- Soft IPL entry point
- Hard IPL entry point
- Disk sector read module
- Diskette sector read module
- NVRAM Write a byte module
- Power-on self test indicator change module
- Speaker beep module.

The table entries are offsets from the start of ROM. The ROM entry point table is accessed by a pointer at location X'C18' in the POST control block.

IPL Types

ROM code provides support for four IPL functions. These are:

1. Normal POR

The normal power-on reset events are performed.

2. Software initiated POR with memory clear

The normal power-on reset events are performed but initiated by a software jump into ROM code.

3. Software initiated soft IPL with no memory clear

The system code is reloaded while preserving the existing machine state as much as possible. Memory is not changed from its preexisting values except for those locations filled by the newly loaded code. This action is initiated by a software jump into ROM code.

4. Keystroke sequence initiated soft IPL with no memory clear

The system code is reloaded while preserving the existing machine state as nearly as possible. Memory is not changed from its preexisting values, except for those locations filled by the newly loaded code. This action is initiated by the keyboard adapter activating the system processor reset signal.

ROM supports these four functions by allowing ROM execution to begin at either of three entry points. These are:

- POR entry by means of the system processor POR vector. Operation from this point will test for item four above and jump to the soft IPL entry if the keystroke sequence was the source.
- Hard IPL entry from software to mimic the POR action above when the keystroke sequence was not the cause.
- Soft IPL entry which will merely reload the system software without going through the power-on self tests or memory tests.

The soft entry points cause execution of setup operations to insure that ROM can execute and then transfers control to the IPL controller. The effect is that all ROM power-on self tests are skipped, memory is not reinitialized, and a new IPL load is performed. The only power-on self test run is the NVRAM test. Running this test ensures that the user can redirect the optional IPL table.

To cause a soft IPL to use a specific IPL device, the following actions could be used:

1. Write the specific device number to be used in the IPL to the first position in the NVRAM IPL device order table, insuring good NVRAM CRC.
2. Cause a soft IPL either to the *clears memory* or the *does not clear memory* ROM locations.

Re-IPL Entry Point Requirements

The following requirements must be met for execution from the ROM IPL entry points to proceed without error:

- ROM must be mapped by the ROM specification register in the memory management unit to real address X'800000', which is the address to which ROM maps itself when power is turned on.
- RAM must not be mapped over the ROM addresses in real memory.

Due to the internal operation of the memory management unit, real memory must be specified to be 8M-bytes or less in size before ROM is accessed or an exception will result from the conflicting specification. If RAM is mapped in an invalid manner, ROM execution causes an appropriate error display and operation halts.

- Accesses must be made in real mode since ROM cannot be accessed in translate mode.
- The first 2K-bytes of the POST control block must be present in memory and contain correct values as determined by the power on execution of ROM.

Information Left After Soft IPL

The soft IPL indicator at H'C10' in the POST control block has three significant values.

H'FFFFFFFF' Power on reset or hard IPL in progress.

H'00000001' Hard IPL is complete, soft IPL is permitted.

H'00000000' Soft IPL has been performed since POR or hard IPL.

Restrictions and Conventions for Externally Callable Routines

To use any of the modules in ROM, the proper run time environment must be established. The ROM itself contains an appropriate run time environment. Use this environment if the POST control block is undisturbed and the calling software allocates the proper stack area as expected by the ROM.

The stack limit pointer for ROM execution is saved in location X'C20' in the POST control block. The user must guarantee that all memory between the address stored in this pointer and the address contained in location X'804' is free for ROM stack use. This is typically about 32K-bytes. Also, the contents of location X'C28' must be 0 for the ROM to use its own run time environment during external calls.

Register Conventions

The following is a list of system processor register conventions for external calls to ROM modules on entry to external routines:

Register	Definition
0	Destroyed over the call.
1	Destroyed over the call.
2-7	Contains the parameters in that order. If more than 6 parameters are used, register 7 is the address of a continuation list for more parameters. Registers 6 and 7 are preserved over the call.
8-13	Working registers; values are preserved over the call, but the values are not used in ROM routines.
14	Must point to the stack.
15	Return address to calling routine.
CS	Destroyed over calls.
MQ	Destroyed over calls.

The following is a list of register conventions for external calls to ROM modules on exit from external routines:

Register	Definition
2	The first, or only, value returned by a function.
6-13	Registers that are preserved by the invoked procedure.
14	Stack address is restored by the invoked routine.
15	Return address to caller.
CS	Destroyed over calls.
MQ	Destroyed over calls.

Stack Usage

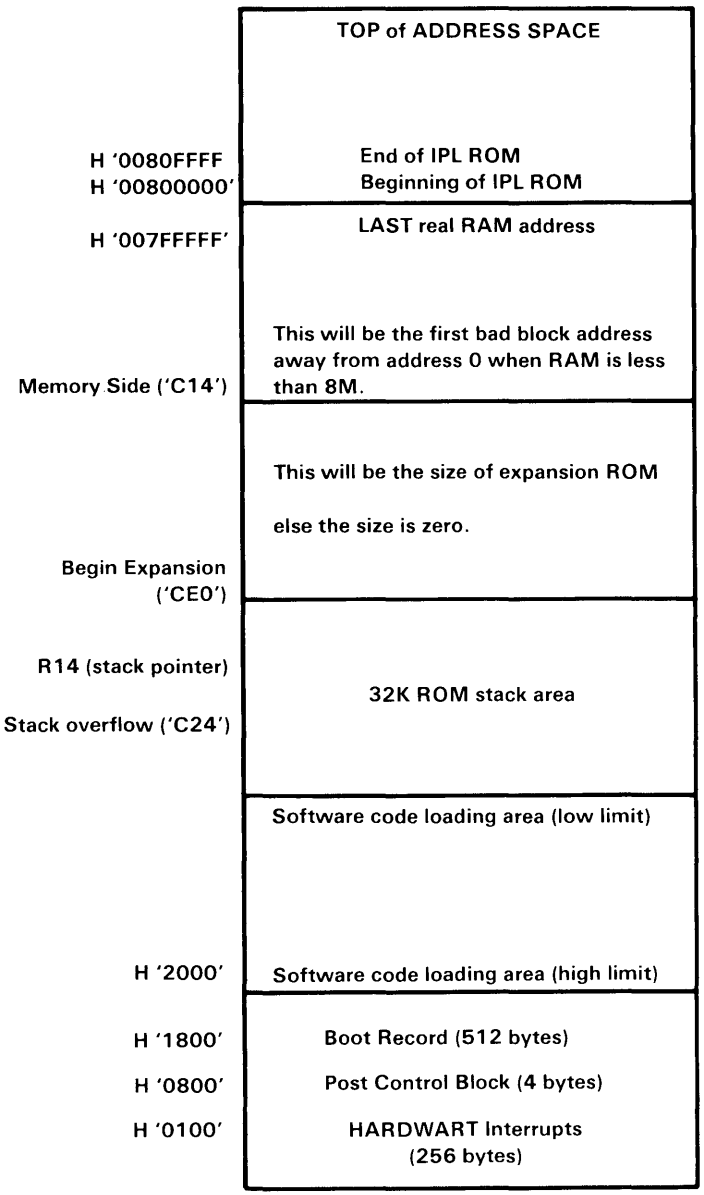


Figure 7-5. RAM and ROM layout during ROM IPL.

If the user wishes to take advantage of the externally callable modules in ROM, a stack area of about 8K (somewhere in RAM) must be provided and R14 must point to the stack area. If this is a new stack, then R14 must be adjusted (to the stack start address - H'40') before the first call to ROM. This allows ROM to immediately save the callers registers in the stack. If the stack has been established this adjustment is not necessary.

Once ROM gives control to the loaded code the 32K ROM stack area will become available to the loaded program code. ROM has been architected to start at address H'0080000' (8M), therefore the highest ROM stack overflow address will be 32K less then 8M (any RAM over over 8M is not mapped to storage while IPL ROM is in control). The lowest ROM stack overflow address will be when the minimum size memory card is installed and the maximum size of expansion ROM is present (maximum size of expansion ROM is 128K).

Example if 1M of RAM and maximum expansion ROM:

```
 1024K memory size
- 128K expansion ROM size
------
 896K is the address of the ROM new stack start
- 32K stack size
------
 864K is the address of the ROM stack overflow
```

Bad blocks in 8M or smaller also cause the stack address to be reduced.

If the PCB location H'C24' is H'0', ROM assumes that the stack is intact as in the ROM IPL environment and will use the IPL ROM stack adjustment and overflow handling module. But if the PCB location H'C24' is not zero (some valid address) then ROM assumes that the loaded code is providing stack adjustment and overflow handling at that address.

Module Linkage Information

The ROM entry point table pointer is located at H'C18' in the POST control block.

Hex Offset	Description
00	Soft IPL entry point
04	Disk Sector Read
08	Diskette Sector Read
0C	Read IPL Device
10	Change LED Indicator
14	Speaker Beep
18	Calculate CRC Value
1C	Hard IPL Entry Point
20	Disk Restore
24	Diskette Restore
28	NVRAM Write a Byte

Figure 7-6. Layout of Offsets in the ROM Entry Table

Soft IPL Entry Point (H'00')

No parameters are required and there is no return to the caller.

Disk Sector Read (H'04')

The CALLING register convention is:

- R2 is a hex value of the starting cylinder
- R3 is a hex value of the starting track
- R4 is a hex value of the starting sector
- R5 is a hex value of the number of sector to read
- R6 is a hex value of the data destination address.

The return register convention is:

- R2 is the Return Code.
  - Return Code 0 signifies successful completion.
  - Return Code 1 signifies failure to perform function.

Note: See 7-5

Diskette Sector Read (H'08')

The calling register convention is:

- R2 is a hex value of the starting cylinder
- R3 is a hex value of the starting track
- R4 is a hex value of the starting sector
- R5 is a hex value of the number of sector to read
- R6 is a hex value of the data destination address.

The return register convention is:

- R2 is the Return Code.
  - Return Code 0 signifies successful completion.
  - Return Code 1 signifies failure to perform function.

Note: See 7-5

Read IPL Device (H'0C')

The calling register convention is:

- R2 is a hex value of the starting cylinder
- R3 is a hex value of the starting track
- R4 is a hex value of the starting sector
- R5 is a hex value of the number of sector to read
- R6 is a hex value of the data destination address.

The return register convention is:

- R2 is the Return Code.
  - Return Code 0 signifies successful completion.
  - Return Code 1 signifies failure to perform function.

Note: See 7-5

Change LED Indicator (H'10')

The calling register convention is:

- R2 is a hex value of the 2 LEDS (lighting emitting diodes). The value is from H'00' to H'FF'. Values of 0 to 9 display the range of 00 to 99 while values H'A' to H'F' may be nonsensical. The value H'FF' is for blanks.
- R3 is a hex value of a delay count before returning. The values are from H'00000000' to H'EFFFFFFF'. The H'00200000' value is about one second delay. There is no return code.

Speaker Beep (H'14')

No parameters are required. The speaker will beep .1 seconds at 2500 Hz. There is no return code.

Calculate CRC Value (H'18')

The calling register convention is:

- R2 is a hex value for the number of sequential bytes used to calculate the CRC value.
- R3 is the hex value of the starting address.

The return register convention is:

- R2 contains the CRC value in hex.

Hard IPL Entry Point (H'20')

No parameters are required and there is no return to the caller. See "IPL Types" on page 7-15.

Disk Restore

The calling register convention is:

- R2 is a hex value of the adapter address. The address for the primary adapter is H'F00001F0'. The address for the secondary adapter is H'F0000170'.
- R3 is a hex value of the maximum cylinders per drive. This value is in cylinder 0, track 0, sector 1 the boot record.

- R4 is a hex value of the maximum tracks per cylinder. This value is in cylinder 0, track 0, sector 1 the boot record.
- R5 is a hex value of the maximum sectors per track. This value is in cylinder 0, track 0, sector 1 the boot record.
- R6 is a hex value of the drive number. This value is H'00000000' or H'00000001'.

The return register convention is:

- R2 is the Return Code.
  - Return Code 0 signifies successful completion.
  - Return Code 1 signifies failure to perform function.

To get the data from the boot record, do a restore sufficient to read cylinder 0, track 0, sector 1. Then read the boot record extract the correct geometry of the disk and do another restore using the boot record data. If there is no boot record, the data is in cylinder 0, track 0, sector 2 in another format different than the boot record data. The format of sector 2 is outside the realm of discussion of ROM and must be referenced in system documentation on the disk configuration record.

Note: See 7-25

Diskette Restore (H'24')

The calling register convention is:

- R2 is a hex value of the adapter address. The address for the primary adapter is H'F00003F2', the address for the secondary adapter is H'F0000372'.
- R3 is a hex value of the drive number. This value is H'00000000' or H'00000001'.

The return register convention is:

- R2 is the return code.
  - Return code 0 signifies successful completion.
  - Return code 1 signifies failure to perform function.

A 360K or 1.2M formatted diskette must be in the drive.

Note: See 7-25

NVRAM Write a Byte (H'28')

The calling register convention is:

- R2 is a hex value of the NVRAM address. The address range is H'F000880E' to H'F000883D'.
- R3 is a hex value of the data byte. The data values are H'00' to H'FF'.

The return register convention is:

- R2 is the return code.
  - Return code 0 signifies successful completion. The byte was written and the NVRAM CRC was updated.
  - Return code 1 signifies failure to perform function. One of the following occurred:
    - Bad address in R2 (user error)
    - The data written to NVRAM did not compare equal when read back.
    - The NVRAM CRC written did not compare equal when read back.

Note: The sector read device must be restored before any attempted use. If this is the IPL device and no other device restore has canceled the IPL restore then no restore is necessary, IPL ROM performed the restore. In order for the ROM code to work harmoniously with the read IPL device module entry when a device other than the IPL ROM device is restored, the PCB entry at H'C04' must be updated to reflect the new read device id.

PCB Location Contents

The following are the contents of the Post Control Block (PCB) by hexadecimal address. The first 2K bytes of the PCB are used by ROM in a soft-IPL or re-IPL and so must be undisturbed by loaded code. Changes to this area will cause unpredictable results when the reset sequence is entered at the keyboard. See "PCB Location Contents Definitions" on page 7-28 for a description of the contents.

Address	Contents
	Memory Test Results
800	Offset from 800 of last defective page address
804-BFF	Defective page addresses on 128K boundaries
	Miscellaneous Data Area
C00	NVRAM indicator
C04	IPL device id
C08	Keyboard id
C0C	Machine type identification word
C10	Soft IPL indicator
C14	Memory size; size of contiguous good RAM from address 0.
C18	ROM entry point table pointer
C1C	Reserved
C20	$START stack pointer save (stack start - H'40')
C24	$XPROL checks ROM stack requirements for fit
C28	Address of $XPROL if other than 0
C2C	ROM stack start restored in soft IPL
C30	Prior IAR save area for time group
C34	Prior ICS save area for time group
C38	Level 1 interrupt module return address
C3C	Flag and RAM specification register save word
C40-C5B	Diskette parameters in PIO diskette (7 words)

C5C-C6B	Diskette dig out reg in PIO disk (4 words)
C6C-C83	PIO disk read module save area (6 words)
C84-CC3	Slot id table (16 words)
CC4-CD3	RAM 128K-bit map of defects (4 words)
CD4	Bad memory address low
CD8	Bad memory address high
CDC	Memory board bad information
CDD-CDE	Reserved
CDF	Memory board size byte
CE0	Begin expansion ROM address if address ¬ = $START (H'C20')
CE4-CEB	Date ROM was linked
CEC-CEF	ROM vestige
CF0	Reserved
CF4-CFF	RAM instructions for 16M set RAM specification register
	POST entry table for POST entries 1 through 48
D00-FFF	This region must be preserved by loaded code
	POST entry table for entries 49 through 176
1000-17FF	For optional adapter tests beyond initial release. This area is transient, paged by VRM; used by diagnostics.

PCB Location Contents Definitions

Memory test results
The first 256 words of the PCB contain defective memory information.

Offset from 800 of last defective page address
This location contains the offset from the start of the PCB of the last defective page found.

Defective page addresses on 128K boundaries
This area contains from zero to 255 addresses of the first byte of 128k byte blocks of memory. If a block is listed here, it contains at least one bad memory location (correctable ECC errors are not considered bad). The addresses are in ascending order, with idle locations containing H'A5A5A5A5'.

Miscellaneous data area
This area contains data, pointers, and other miscellaneous information used in ROM execution.

NVRAM indicator
This word contains a 0 if the CRC value of the NVRAM is correct and a H'00000001' if it is not.

IPL device id
This word contains the storage device number from which the IPL was obtained. Valid values are:

000000D0 disk 0
000000D1 disk 1
000000D2 disk 2
000000D3 disk 3
000000F0 diskette 0
000000F1 diskette 1
000000F2 diskette 2
000000F3 diskette 3

Keyboard id
This word contains the keyboard ID code obtained when the keyboard initial test was run. Valid values are:

0000BFB0 - The G keyboard
000081Fx - Manufacturing IPL id
FFFFFFFF - No keyboard id found

Machine id
This word shows whether the machine is a IBM 6150 or a IBM 6151. If it contains 0, the machine is a IBM 6150; H'FFFFFFFF' is the IBM 6151.

Soft IPL indicator
Shows whether a soft IPL has been done since POR. If checked and and reset appropriately by loadable code, it can be used to determine when a soft IPL has been done.

H'FFFFFFFF' = Power On Reset or Hard IPL in progress.
H'00000001' = normal POR completed.
H'00000000' = entered soft IPL.

Memory size
This word contains the memory size in bytes of the contiguous good memory starting from location 0. This may not be the largest contiguous block of good memory; the proper way to determine the map of good memory is by using the memory board size byte described below in combination with defective page addresses.

ROM entry point table pointer
This is the table of entry points address to ROM for external use. It is normally set to point to this table at H'00800008', which is the table position in ROM.

Reserved
Not used, value is not guaranteed and may change.

$START stack pointer save
Saved pointer to the ROM start stack area - H'40'

$XPROL checks ROM stack requirement for fit
Address of Stack overflow.

Address of users $XPROL if other than 0
Address set by the user to point to their $XPROL if they do not use the IPL ROM stack area on external calls. see "Stack Usage" on page 7-19 for more information.

ROM stack start restored in soft IPL.
Same as $START above.

Prior IAR save area for time group
Temporary data area used in timer functions; may contain any value.

Prior ICS save area for time group
Temporary data area used in timer functions; may contain any value.

Level 1 interrupt module return address
Temporary data area used in timer functions; may contain any value.

Flag and RAM spec word

Contains various flags and the save value of the RAM specification register.The values are:

Bits 0-7	PC mono slot reset mask
Bits 8-10	Not used; always 0.
Bit 11	When 1, PC mono adapter is present
Bit 12	When 1, board in slot C is faulty
Bit 13	When 1, board in slot D is faulty
Bit 14	Keyboard interrupt occurred
Bit 15	Timer interrupt occurred
Bits 16-31	RAM specification register save area for soft IPL

Diskette parameters in PIO diskette (7 words)

Data save area used in diskette IPL. Value is not guaranteed.

Diskette dig out reg in PIO diskette (4 words)

Data save area used in diskette IPL. Value is not guaranteed.

PIO disk read module save area (6 words)

Data save area used in disk IPL. Value is not guaranteed.

Slot id table (16 words)

This 64-byte area contains the adapter types found in each slot. ROM execution fills in only the disk/diskette adapter. Loadable POSTs fill in the adapter types found during their execution. It is required that the slot determination algorithms used in the loadable POSTs not reset the disk/diskette adapter while finding their respective adapters. The locations have the following format:

H'C84 - C8B' List of adapters in slot 1.
H'C8C - C93' List of adapters in slot 2.
H'C94 - C9B' List of adapters in slot 3.
H'C9C - CA3' List of adapters in slot 4.
H'CA4 - CAB' List of adapters in slot 5.
H'CAC - CB3' List of adapters in slot 6.
H'CB4 - CBB' List of adapters in slot 7.
H'CBC - CC3' List of adapters in slot 8.

The adapters are listed one per byte in the order found. For example, the slot table entries showing one primary disk/diskette board in slot one would be:

'C84'--5253FFFF FFFFFFFF

The adapter types for disk and diskette being 52 and 53 respectively. ROM execution should show all slots empty except the slot containing the disk/diskette adapter.

RAM 128K bit map of defects (four words)
This four word area contains a map of the entire 16M real address space. Each bit represents one 128K block of memory with a 0 meaning good and a 1 being bad. For 2M of installed good memory, the bit map is:

H'CC4'- 0000FFFF FFFFFFFF FFFFFFFF FFFFFFFF

Bad memory address low
This is the address of the first byte of memory showing a failure including any correctable ECC errors. It is used primarily for manufacturing tests.

Bad memory address high
This is the address of the last byte of memory showing a failure including any correctable ECC errors. It is used primarily for manufacturing tests.

Memory board bad information
This byte contains information on which memory board has errors in its memory space. The information is bit coded with a 1 bit representing a bad board. Values found will be:

B'00000000' - no bad boards
B'00000001' - board in slot C contains failures
B'00000010' - board in slot D contains failures
B'00000011' - both boards contain failures

Reserved Bytes
These two bytes are reserved for future memory test expansion and are now set to all zeros.

Memory board size byte
This byte contains the value found in the system board memory configuration register. See "Memory Configuration Register" on page 5-73.

Begin expansion ROM address if address ¬ = $START(H'C20').
This word is the address of $Start or Expansion ROM start address when expansion ROM is present.

Date ROM was linked
This is ROM link date, when the ROM object code was created.

ROM VESTIGE.
For diagnostic use.

Reserved
This area is reserved for future expansion of the diagnostics.

RAM instructions for 16M set RAM specification register
This area is used by soft IPL ROM when the RAM specification register is set greater than 8M.

POST entry table for POST entries 0 though 48
This area contains the four word POST entries for POSTs 0 through 48. All of these locations are reserved for present and future loadable POSTs.

POST entry table for entries 49 through 175
This area contains the four word POST entries for POSTs 0 through 48. All of these locations are reserved for present and future loadable POSTs. This area may be paged or modified by loaded code. It is used by diagnostics and will be used by future loadable POSTs. Only entries numbered 5,6,7,8,12,13,14,15,16, and 21 are modified by ROM execution. The others are left at their initialization state of(using #1 located at D00 as an example)

H'D00' - 00000001 FFFFFFFF FFFFFFFF 88FFFFFF

Display Codes

The following is a list of the IPL ROM display codes.

00 - Initializing PC Mono adapter
0c - Initializing PC Mono adapter (Advanced Processor)
01 - ROM CRC error
1c - ROM CRC error (Advanced Processor)
02 - Error testing for soft IPL
03 - Memory test- 1st 128K-bytes
3c - Memory test- 1st 128K-bytes (Advanced Processor)
04 - Processor board bad
4c - Advanced Processor board bad
05 - Processor board or memory bad
5c - Advanced Processor board or memory bad
07 - IOCC test
7c - IOCC test (Advanced Processor)
08 - Processor board failed in IOCC test
8c - Advanced Processor board failed in IOCC test
09 - Keyboard adapter
10 - System timer (RTC)
11 - Interrupt controller test
12 - DMA test
13 - Serial ports
14 - Fixed Disk and Diskette Adapter Board 1
15 - Fixed Disk and Diskette Adapter Board 2
16 - Fixed Disk and Diskette Adapter Board 1
17 - Fixed Disk and Diskette Adapter Board 2

18 - Expansion ROM in control
19 - Manufacturing IPL
20 - NVRAM CRC
21 - NVRAM order IPL
22 - ROM default order IPL
23 - No IPL device found
25 - Soft IPL check failure condition
26 - Attempting soft IPL
27 - Bootable code won't fit in available RAM
29 - Loadable code has been started
88 - Machine check stop
89 - Unexpected program check or machine check
96 - Memory board in slot C bad
97 - Memory board in slot D bad
98 - Both memory boards bad
99 - Keylock locked

Boot Record

The following diagram shows the IPL boot record for disk and diskette.

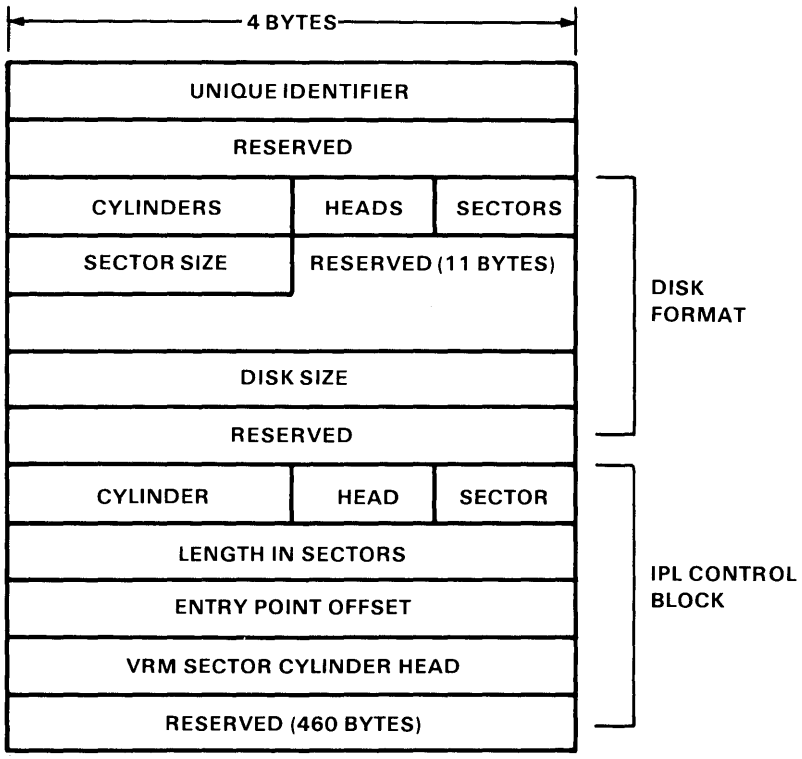


Figure 7-7. IPL ROM Boot Record Format

The following is an explanation of the fields in the boot record.

Unique Identifier	H'C9C2D4C1'
Reserved fields	All reserved fields are set to zero.
Number of cylinders	The number of cylinders in the disk used; zero for diskette.
Number of heads	The number of heads in the disk used; zero for diskette.
Sectors/Track	The number of tracks in the disk used; zero for diskette.

Sectors size	The size of one sector in bytes.
Formatted disk size	
Cylinder	The cylinder containing the start of the loaded code.
Head	The head number to be used to read the loaded code.
Sector	The first sector of the loaded code.
Length in sectors	The length of the loaded code in sectors.
Entry Point Offset	The offset from the start of the loaded code to the first instruction.
VRM cylinder head sector	The address of the VRM minidisk in cylinder, head, sector format.

Section 8. System Compatibility

CONTENTS

About this Section

This section describes the RT PC system compatibility. RT PC is upward compatible only. The information provided in this section is for RT PC only.

I/O Channel Compatibility

System Board

The RT PC system board uses an Input/Output Channel Controller (IOCC) to drive its I/O channel. The IOCC simulates the PC Family I/O channel and allows some PC I/O adapters to operate. However, there are some differences in the signal definitions.

Some system board slots on the IBM RT PC have a 40-pin connector in addition to the 62-pin connector.

Although some may not operate correctly, all PC and PC AT options and adapters can be plugged into the RT PC system board. The RT PC adapters that use all 40 pins will not plug into PC AT system boards.

I/O Channel Signals

- Addresses. All address lines on the RT PC I/O Channel are latched early in the I/O cycle. This is done for performance reasons.
- Address lines SA17 - SA19. These lines must be driven by alternate controllers with the same data as LA17 - LA19.
- BALE. The 'BALE' signal is held high at all times because the addresses are all latched by the I/O channel.
- Oscillator. The 14.31818 MHz. oscillator is not synchronous with the system clock.
- Clock. The clock line runs at a frequency of 4.77 MHz.
- Refresh is done in bursts of 5 to 16. Refresh operations for channel attached RAM is discussed in Section 6 of this manual.

Unique Pin Assignments

The IBM RT PC System Boards each have one unique slot for the optional 286 Coprocessor board. This slot will not support memory refresh or DMA Channel 7. DMA channel 8 arbitration is different than DMA channels 0 through 3 and 5 through 7. The arbitration for DMA channel 8 is discussed in the Coprocessor arbitration section of this manual. The following chart shows the standard and unique pin assignments.

Pin	I/O Slot Standard	I/O Slot Coprocessor
B19	REFRESH	+ SPK DRV
D14	DACK 7	- DACK 8
D15	DRQ 7	+ DRQ 8

RT PC Compatibility With IBM PC Products

The IBM RT PC supports the following options and adapters:

- IBM RT PC Streaming Tape Drive Adapter
- PC Network Adapter
- 4-Port Asynchronous RS-232C Adapter
- 4-Port Asynchronous RS-422A Adapter
- Serial/Parallel Adapter
- Monochrome Display and Printer Adapter
- Enhanced Graphics Adapter
- RT PC Coprocessor
- AT 512 KB Memory Expansion Option
- Prototype Adapter
- 3278/79 Emulation Adapter
- Fixed Disk and Diskette Adapter
- Advanced Color Graphics Adapter
- Advanced Monochrome Display Adapter

- RT PC 5080 Attachment Adapter
- RT PC S/370 Host Interface Adapter
- IBM 5080 Peripheral Adapter
- Multiprotocol Adapter
- Extended Monochrome Graphics Display Adapter
- Token Ring Adapter
- Base Band Adapter
- ESDI Magnetic Media Adapter
- Extended ESDI Magnetic Media Adapter
- Small Computer System Interface Adapter
- Megapel Display Adapter

The IBM RT PC does not support any other options or adapters. It does not support any other keyboard or locator other than those sold with the product.

Note: This information is accurate at the time of publication.

Keyboard Compatibility

The IBM RT PC keyboards include a 101 key unit for U. S. machines and various 102 key units for world trade machines. It is not plug or electrically compatible with any of the other IBM PC family keyboards.

Locator Compatibility

The IBM RT PC locator is not compatible with any other IBM PC family mouse hardware.

PC Software Compatibility

PC software compatibility is addressed in the *IBM RT PC 286 Coprocessor Technical Reference* manual.

Software Compatibility

The IBM PC Enhanced Graphics Adapter and the 3278/79 Emulation Adapter both use interrupt level 9. The 3278/79 Emulation Adapter must use interrupt level 9 to operate. Some EGA applications may cause interrupt level conflicts. Interrupt level 9 cannot be enabled by the EGA adapter application if you wish to operate the EGA adapter and the 3278/79 Emulation Adapter concurrently. See the IBM RT PC 286 Coprocessor Technical Reference Manual for further information on compatibility with EGA adapter and 3278/79 Emulation Adapter.

Section 9. User Input and Output Devices

CONTENTS

About this Section

This section contains information about the keyboard and locator input devices and operator panel output device used by the IBM RT PC system.

Keyboard

The keyboard has a permanently attached cable. The cable's connector has a special overmolding that keys it to a connector at the rear of the keyboard. This shielded six-conductor cable has power (+5 Vdc), ground, two bidirectional signal lines, and two speaker lines. The cable is approximately 2.75 meters long.

The keyboard uses a capacitive technology with a microcomputer performing the keyboard scan function. The keyboard has two tilt positions for operator comfort (7 and 15 degrees) and a speaker for use by the system for generating tones.

The keyboard has 101 keys arranged in four major groups. The central portion of the keyboard is a standard typewriter layout. At the top are 12 user software defined function keys and four predefined function keys. To the immediate right of the central portion are 10 cursor control keys and on the far right is a numeric pad.

The keyboard interface allows system software to have maximum flexibility in defining certain keyboard operations. This is accomplished by having the keyboard return a unique scan code for each key. All keys are individually programmable to recognize the following conditions:

- Make only
- Make/break
- Repeat
- Repeat make/break
- Double rate repeat
- Double rate repeat-make/break.

Note: Repeat and repeat-make/break keys cannot be mixed on any keyboard.

The repeat rate is programmable from 3 to 40 characters per second and repeat delay is programmable from 300 to 600 microseconds. A break code is formed by prefixing the scan code with X'F0'.

The microcomputer in the keyboard performs several functions, including a self test at power-on or when requested by the system. Additional keyboard functions are:

- Keyboard scanning
- Buffering up to four key scan codes
- Maintaining bidirectional serial communications with the system unit
- Executing the hand-shake protocol required by each scan code transfer.

Keyboard Command Set

Hex Command	Description
FF	Reset: Perform power-on check and report.
FE	Resend: Resend last byte.
EE	Echo: Respond with X'EE'.
FD	Set make: Set the subsequent list of keys to make type keys.
FC	Set make/break: Set the subsequent list of keys to make/break.
FB	Set repeat or repeat-make/break: Set the subsequent list of keys to repeat or repeat-make/break.
FA	Set repeat double-rate or repeat-make/break double-rate: Set the subsequent list of keys to repeat double-rate or repeat-make/break double-rate.
F9	Set all make: Set all the keys to make.
F8	Set all make/break: Set all keys to make/break.
F7	Set all repeat or repeat-make/break: Set all keys to repeat or repeat-make/break.
F6	Set default: Set all keys to repeat-make/break.
F5	Default disable: Set all keys to repeat-make/break and disable scanning.
F4	Enable: Start scanning.
F3	Set repeat rate/delay: Set the repeat delay to 300 ms. + 100 ms. times the binary value of bits 1 and 2 (bit 0=MSB). Set the repeat rate according to the following: Period = (8 + A) x 2^B x .00313 s. A = Binary value of bits 5, 6, 7 B = Binary value of bits 3, 4.
F2	Read ID: Respond with X'BFB0'.
F1	Toggle keytype menu: Change to the other menu. Alternate is the default.

Standard Menu	Alternate Menu
Make Only	Make Only
Make/Break	Make/Break
Repeat	Repeat-Make/Break
Double Rate Repeat	Double Rate Repeat-Make/Break

ED Set LED indicator: The LED indicators are set according to bits 5, 6, 7 of the subsequent byte.

Bit	LED
5	Num lock
6	Caps lock
7	Scroll lock

Other Keyboard Outputs

Hex Outputs	Description
FE	Resend: Output by keyboard following receipt of a frame with incorrect parity or stop bit.
FC	Command Reject: Output by keyboard following receipt of an invalid command or parameter.
FA	ACK: Output by keyboard following receipt of any valid input other than Echo or Resend.
00	Overrun: Placed in last position of the 4 position keystroke queue to indicate overflow of the queue.
F0	Break code: Prefixed to the scan code of a make/break key to indicate break of the key.
AA	BAT completion code: Output following successful completion of the self-tests. Any other output indicates failure of the tests.
EE	Echo: Output in response to 'Echo' command.
BF B0	Keyboard ID: Output in response to 'Read ID' command or following BAT completion code.

The following pages have figures that show the keyboards, the scan codes, and the keyboard interface connector specifications.

Keyboard Layout

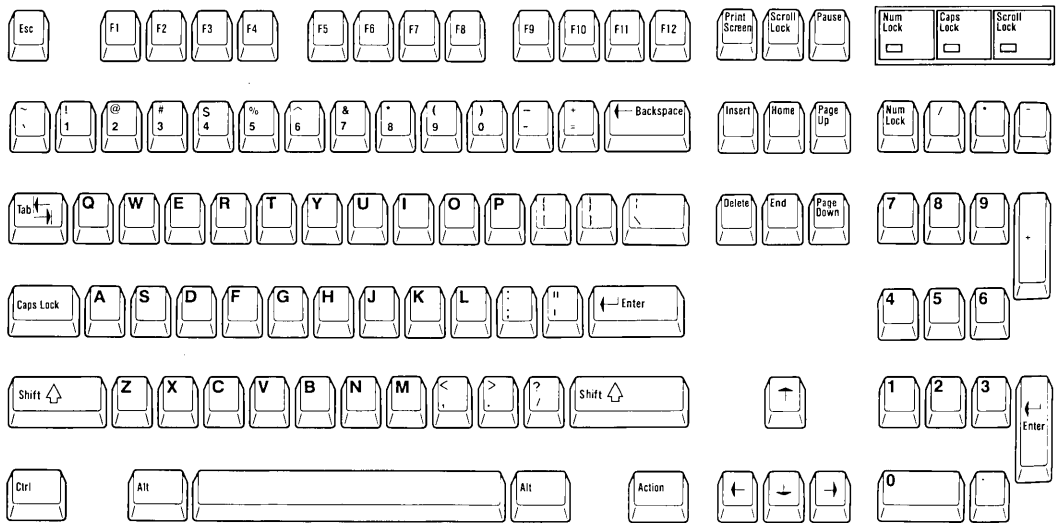


Figure 9-1. US Keyboard

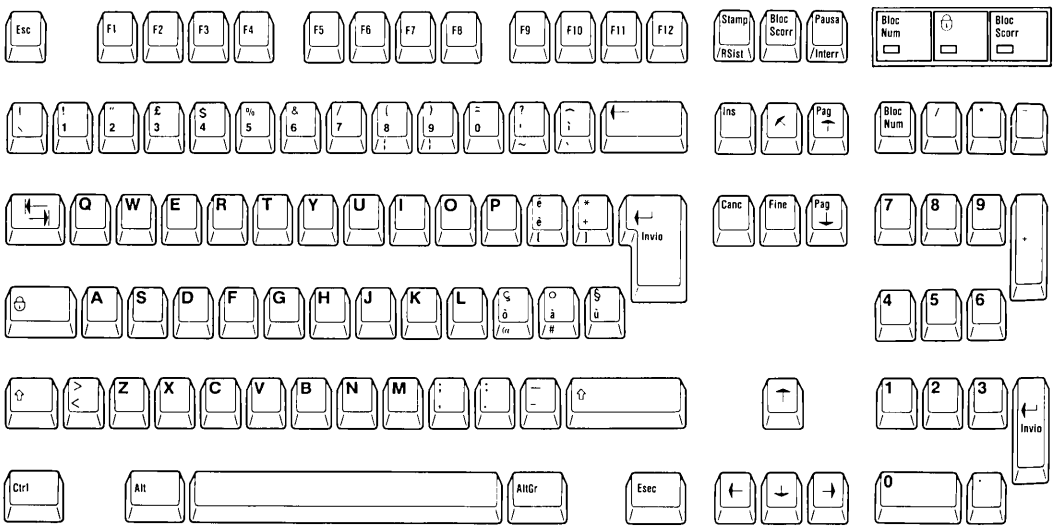


Figure 9-2. Italy Keyboard

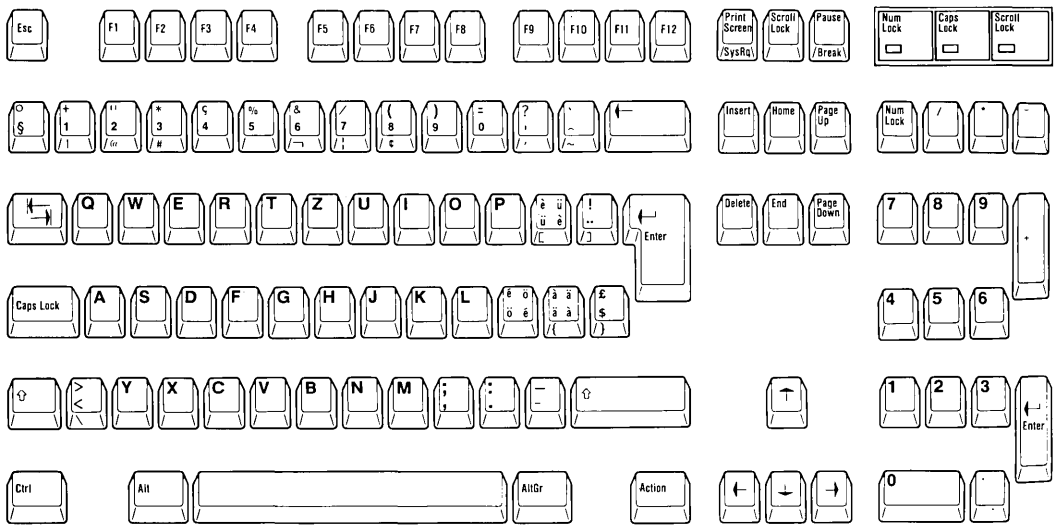


Figure 9-3. Switzerland Keyboard

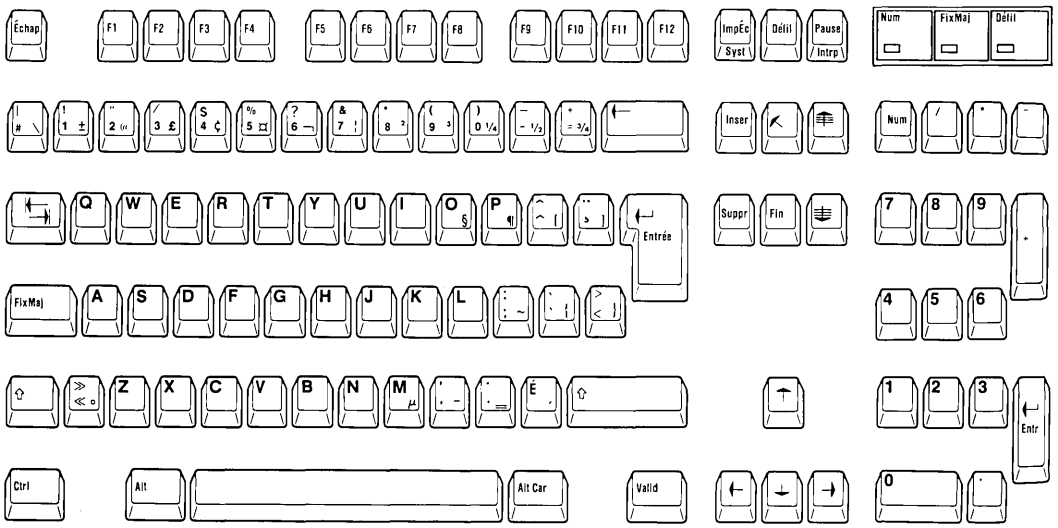


Figure 9-4. Canada (French) Keyboard

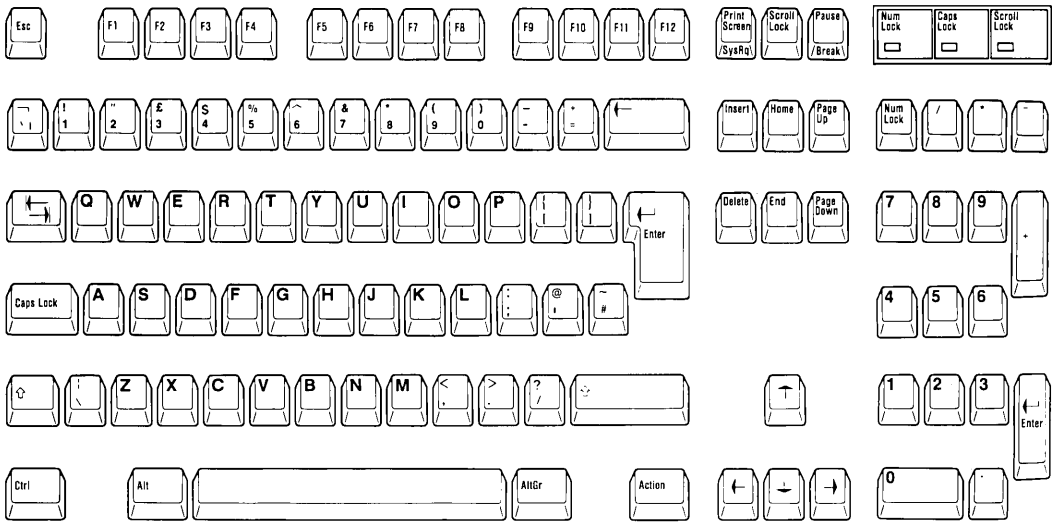


Figure 9-5. United Kingdom (English) Keyboard

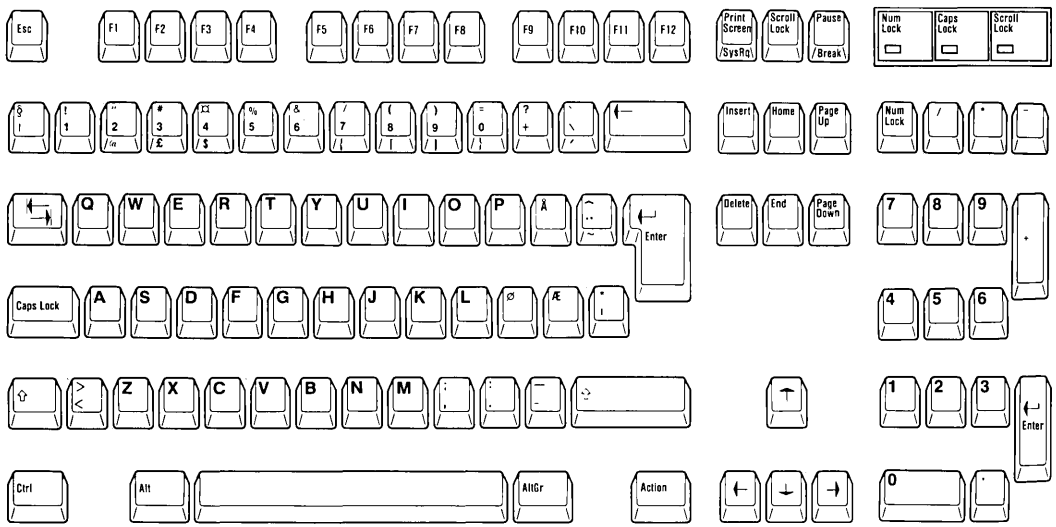


Figure 9-6. Norway Keyboard

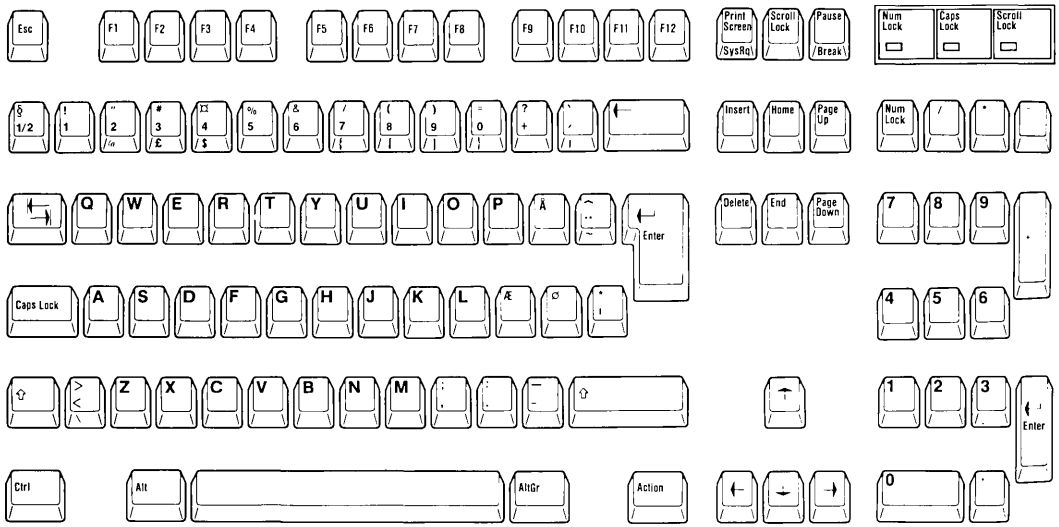


Figure 9-7. Denmark Keyboard

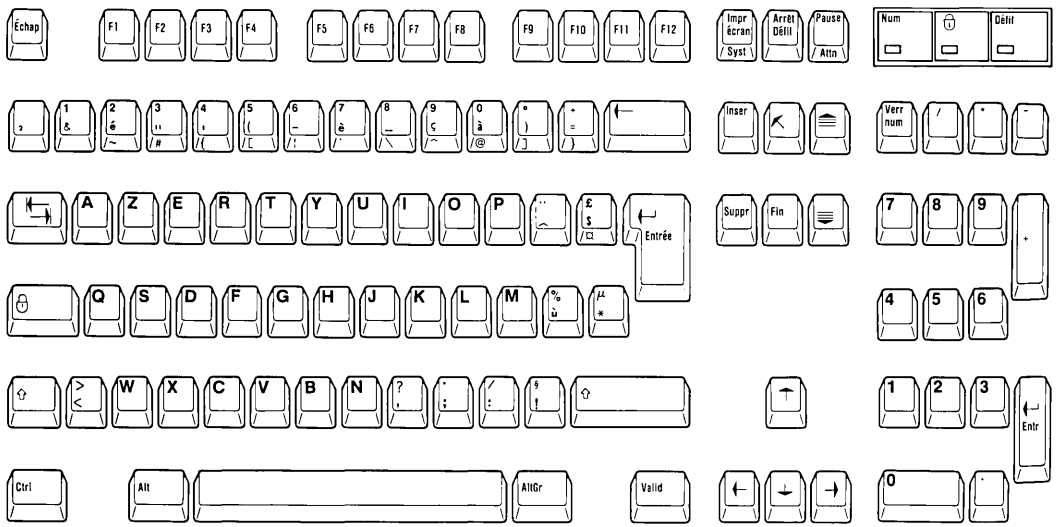


Figure 9-8. France Keyboard

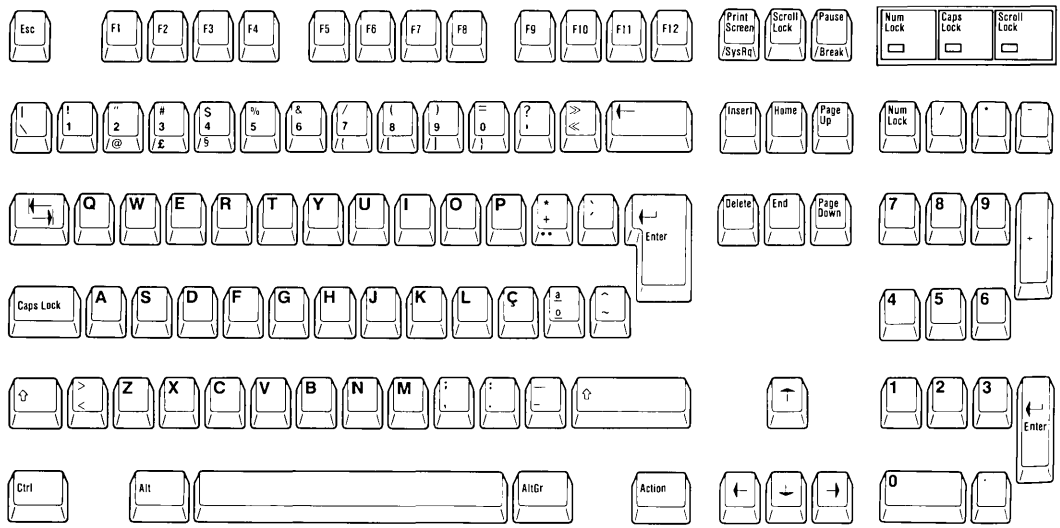


Figure 9-9. Portugal Keyboard

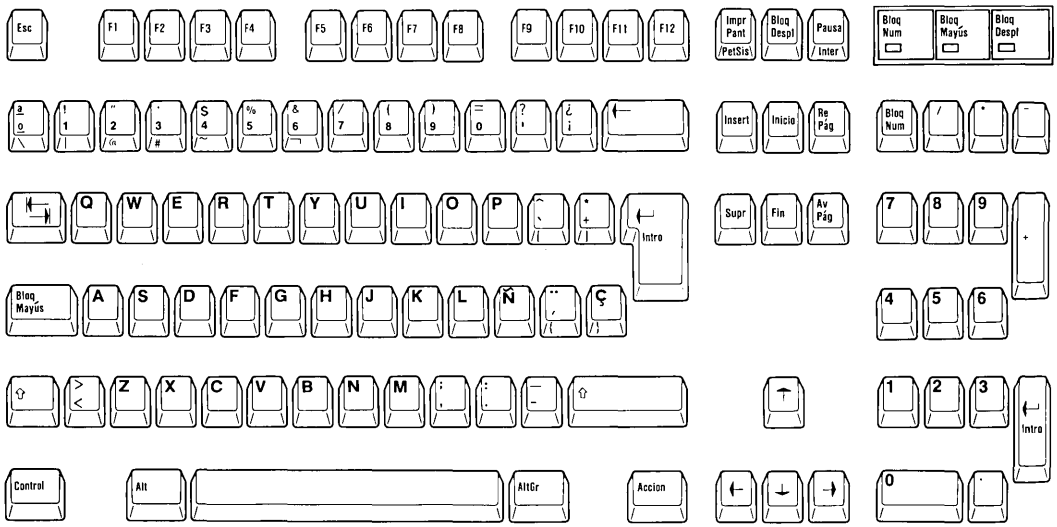


Figure 9-10. Spain Keyboard

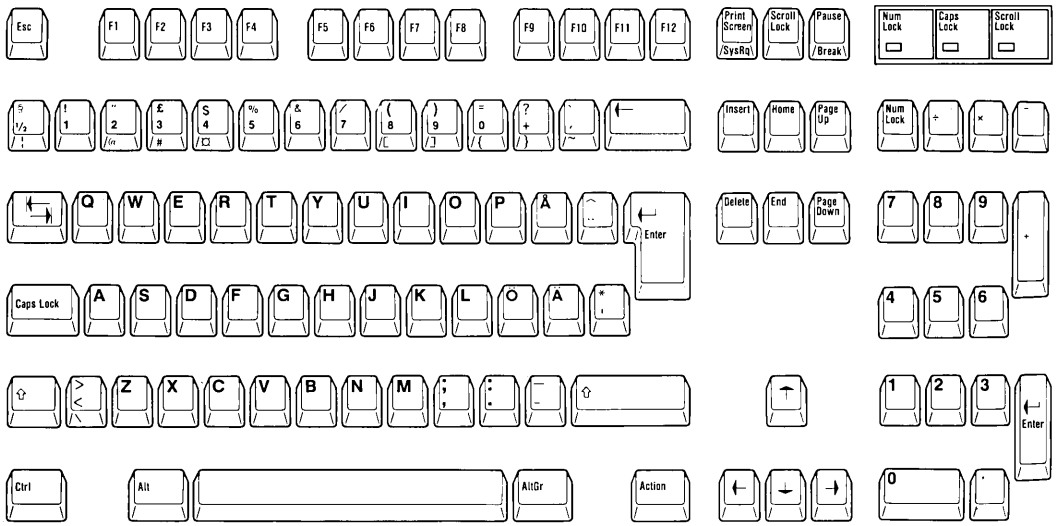


Figure 9-11. Sweden and Finland Keyboard

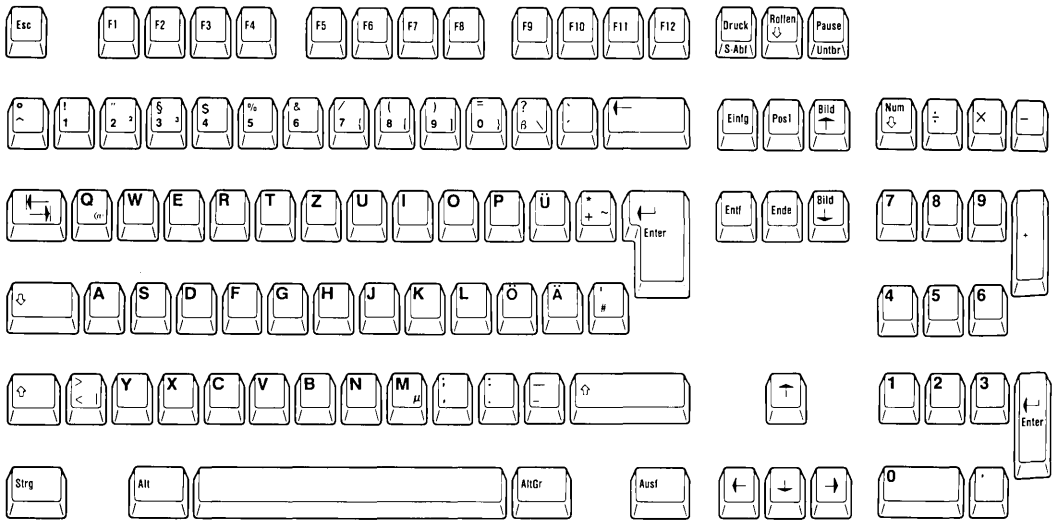


Figure 9-12. Germany Keyboard

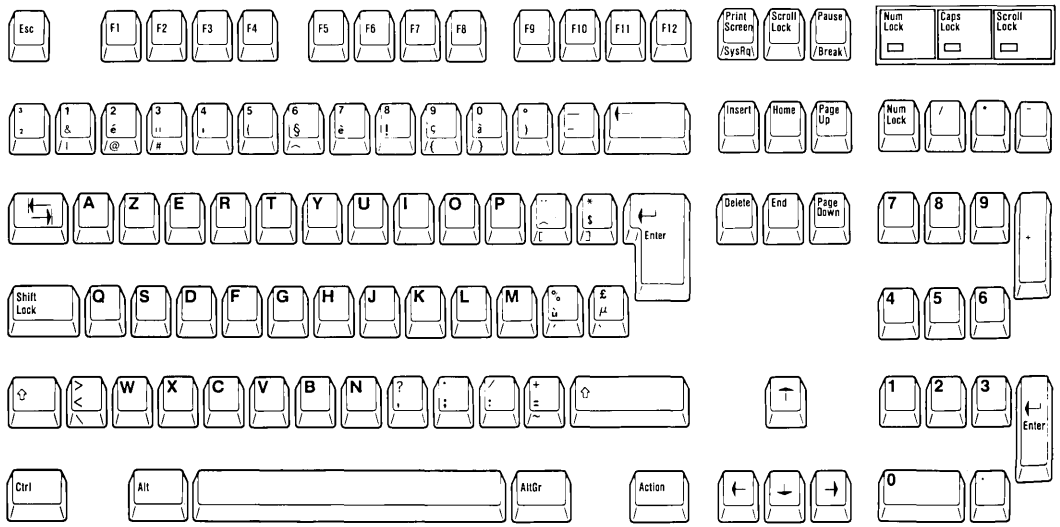


Figure 9-13. Belgium (Dutch/French) Keyboard

Key Position Layout

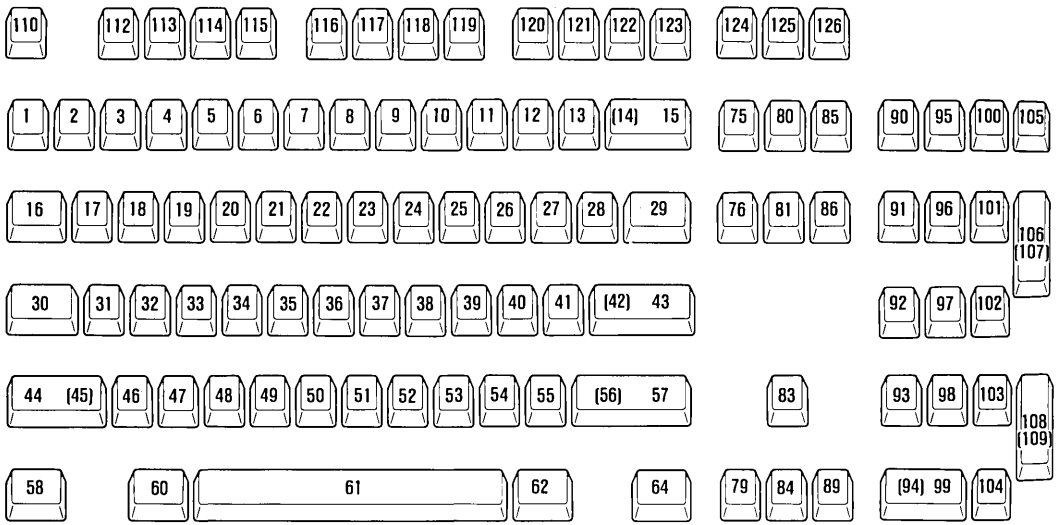


Figure 9-14. US 101 Key Position Layout

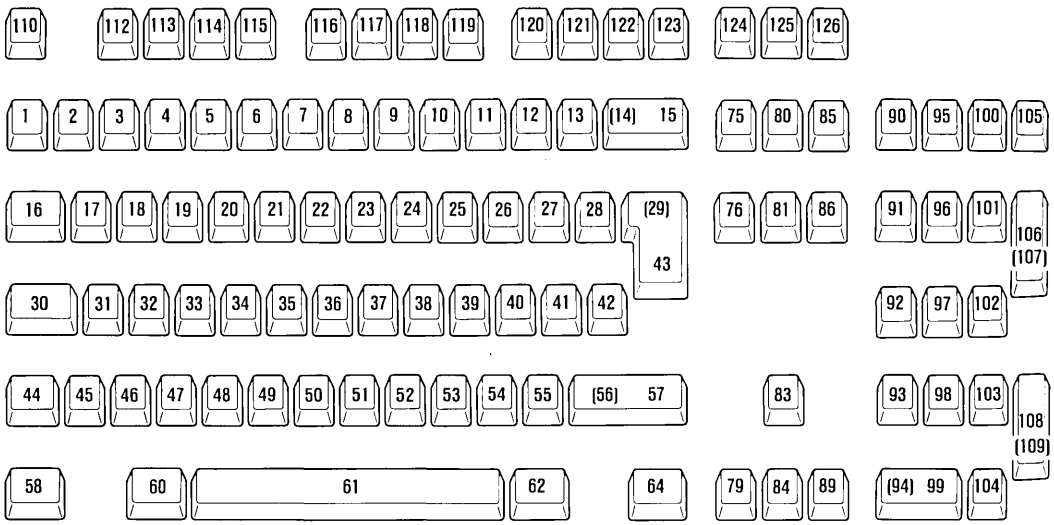
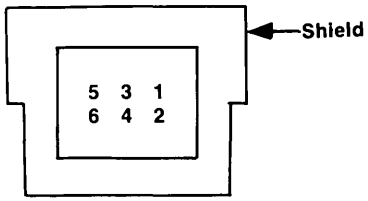


Figure 9-15. WT 102 Key Position Layout

Key Pos	Scan Code	Key Pos	Scan Code	Key Pos	Scan Code	Key Pos	Scan Code
1	0E	33	23			97	73
2	16	34	2B			98	72
3	1E	35	34			99	70
4	26	36	33			100	7E
5	25	37	3B			101	7D
6	2E	38	42			102	74
7	36	39	4B			103	7A
8	3D	40	4C			104	71
9	3E	41	52			105	84
10	46	42	53			106	7C
11	45	43	5A	75	67		
12	4E	44	12	76	64	108	79
13	55	45	13				
		46	1A			110	08
15	66	47	22	79	61		
16	0D	48	21	80	6E	112	07
17	15	49	2A	81	65	113	0F
18	1D	50	32			114	17
19	24	51	31	83	63	115	1F
20	2D	52	3A	84	60	116	27
21	2C	53	41	85	6F	117	2F
22	35	54	49	86	6D	118	37
23	3C	55	4A			119	3F
24	43					120	47
25	44	57	59	89	6A	121	4F
26	4D	58	11	90	76	122	56
27	54			91	6C	123	5E
28	5B	60	19	92	6B	124	57
29	5C	61	29	93	69	125	5F
30	14	62	39			126	62
31	1C			95	77		
32	1B	64	58	96	75		

Figure 9-16. Scan Codes Table

Connector Specifications



PIN	SIGNAL
1	+5 Volts
2	Data
3	Speaker
4	Speaker
5	Ground
6	Clock

Figure 9-17. Keyboard Cable Connector

Locator

The locator uses a rubber coated ball and two optical encoders to indicate the *x* and *y* movement to the system. Two push button switches are located on the locator and their states are transmitted directly to the system. The locator is connected to the system with a thin 2.5 meter long cable whose connector has a special overmolding to key it to a connector on the rear of the system unit. The ball is removable for cleaning.

Operational Characteristics

Resolution	Programmable to 25, 50, 100, or 200 counts per inch. Default=100.
Sampling Rate	Programmable to 10, 20, 40, 60, 80, or 100 reports per second. Default=100.
Data Modes	Stream (default), remote.
Scaling	Linear (default), exponential.
Power	+12 Vdc, ±10%, < 100 mA
	+5 Vdc, ±5%, < 300 mA
	-12 Vdc, ±10%, < 100 mA
Protocol	RS232C, 9600 baud, asynchronous full duplex, stop bit = 1, odd parity.

Operation Modes

Mode	Description
Reset	At power on or on receipt of a Reset command, the locator performs a self test and transmits a four byte power on error report as follows: X'FF', X'08', X'00', X'00'. The following defaults are set: sampling rate = 100 reports per second, linear scaling, stream mode, 100 counts per inch, and disabled.
Stream	In this mode the locator transmits a data report to the system, if a switch is pressed or released or if at least one count of movement has been detected. The maximum rate of transfer is the programmed sample rate. No transmissions occur while the locator is motionless unless a switch is operated, in which case the incremental movement report is zero.
Remote	In this mode the locator transmits data only in response to a read data command.
Wrap	In this mode any byte of data sent by the system, except X'01' or X'0F', is returned by the locator.

Commands

Hex Command	Description
01	Reset: Causes the locator to enter the reset mode.
06	Read Configuration: Causes the locator to transmit one byte: X'20'.
08	Enable: Allows the locator to begin transmissions if it is in stream mode.
09	Disable: Used in stream mode to stop transmissions from the locator. The locator responds to all other commands while disabled. If the locator is in stream mode, it must be disabled prior to sending it any command that requires a response from the locator.
0B	Read X, Y Data: Used in remote mode to cause the locator to transmit one data report.
0E	Set Wrap Mode: Puts the locator in wrap mode.
0F	Reset Wrap Mode: Resets wrap mode.
78	Set Exponential Scaling: Exponential scaling modifies the X and Y data reported by the following transform: Transmitted value = Integer$\{2^{[(V \times S)/R.]-3}\}$ Where: **V** = the current incremental X or Y value **S** = sampling rate in samples/second **R** = resolution in counts/inch **Integer** = integer portion of the expression in brackets Exponential scaling is only performed in stream mode.
6C	Reset Exponential Scaling: Restores linear scaling.

73 Query Status: Causes the locator to send the following 4-byte status report:

Byte 1	Bit 0	Always = 1
	1	Always = 1
	2	Always = 1
	3	Always = 0
	4	0 = Exponential scaling; 1 = Linear scaling
	5	0 = Enabled; 1 = Disabled
	6	0 = Stream mode; 1 = Remote mode
	7	Always = 1
Byte 2	Bit 0	1 = Right key depressed
	1	Always = 0
	2	1 = Left key depressed
	Bits 3-7	Always = 0
Byte 3	Bits 0-6	Current resolution setting (LSB = 6)
	7	Always = 0
Byte 4	Bits 0-7	Current sampling rate (LSB = 7)

'8A' 'XX' Set Sampling Rate: Sets the sampling rate to the value indicated by byte 'XX' given below:

Second Byte 'XX'	**Sample Rate**
X'0A'	10/sec.
X'14'	20/sec.
X'28'	40/sec.
X'3C'	60/sec.
X'50'	80/sec.
X'64'	100/sec.

'8D' '00' Set Stream Mode: Sets stream mode.

'8D' '03' Set Remote Mode: Sets remote mode.

'89' 'XX' Set Resolution: Sets the resolution to the value specified by byte 'XX' given below:

Second Byte 'XX'	Resolution
X'00'	200/inch
X'01'	100/inch
X'02'	50/inch
X'03'	25/inch

Data Report

The data report format shown below is valid for both stream and remote modes:

Byte 1	Bits 0-7	Always = 00001011 ; X'0B'
Byte 2	Bit 0	1 = Right key depressed
	1	Always = 0
	2	1 = Left key depressed
	3, 4	Always = 00
	5	X data sign bit (1 = negative value)
	6	Y data sign bit (1 = negative value)
	7	Always = 0
Byte 3	Bits 0-6	X data (Bit 6 = LSB)
	7	Always = 0
Byte 4	Bits 0-6	Y data (Bit 6 = LSB)
	7	Always = 0

Negative values of X, and Y data are expressed in twos complement. If greater than 127 counts are counted in a sample interval, 127 is reported at the end of the interval. After a transmission the accumulators are set to zero.

Error Handling

If the locator detects an error during a self-test, it transmits, if possible, the power on error report with the error flag bit set (bit 6 of byte 2).

If the locator detects an invalid command, parity, or framing error on any transmission from the system, it transmits the power-on error report with the error flag bit set within 25 ms.

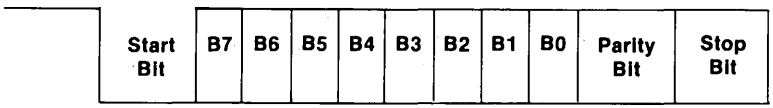
The locator responds to commands requiring a response within 25 ms.

No more than four nonresponse commands may be sent to the locator in succession.

In wrap mode, the host transmission will be returned by the locator exactly as received even if it contains a parity error.

Data Frame

The data frame is as follows:



Voltage Interchange Information

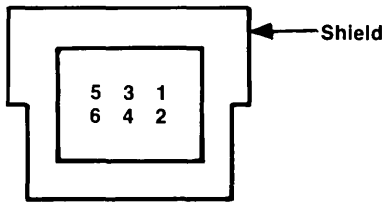
During the transmission of data, the marking condition denotes the binary state 1 and spacing condition denotes the binary state 0.

For interface control circuits, the function is on when the voltage is more positive than +3 Vdc with respect to signal ground, and is off when the voltage is more negative than -3 Vdc with respect to signal ground.

Interchange Differential Voltage (A-B)	Binary State	Signal Condition	Interface Control Function
Positive Voltage	Binary 0	Spacing	= On
Negative Voltage	Binary 1	Marking	= Off

Figure 9-18. Locator Signal Levels

Connector Specifications



PIN	SIGNAL
1	Ground
2	Transmit
3	+12 Vdc
4	-12 Vdc
5	+5 Vdc
6	Receive

Figure 9-19. Locator Cable Connector

Operator Panel

The operator panel is located on the front of the system unit. The panel consists of the keylock, power-on light and two-digit display.

The keylock has both locked and unlocked positions. If you turn on the power switch while the keylock is in the locked position, the system stops, and the two-digit display indicates 99. You cannot enter keystrokes or perform any other operations until the keylock is turned to the unlock position. When you turn the keylock to the unlocked position, the power-on procedure and program loading continue. The keylock may be locked any time after the program is loaded and operating correctly. The power-on light is green, indicating that the power supply is functioning properly.

The two-digit display provides information about the progress of the internal Power On Self Tests (POSTs) and errors encountered in software loading.

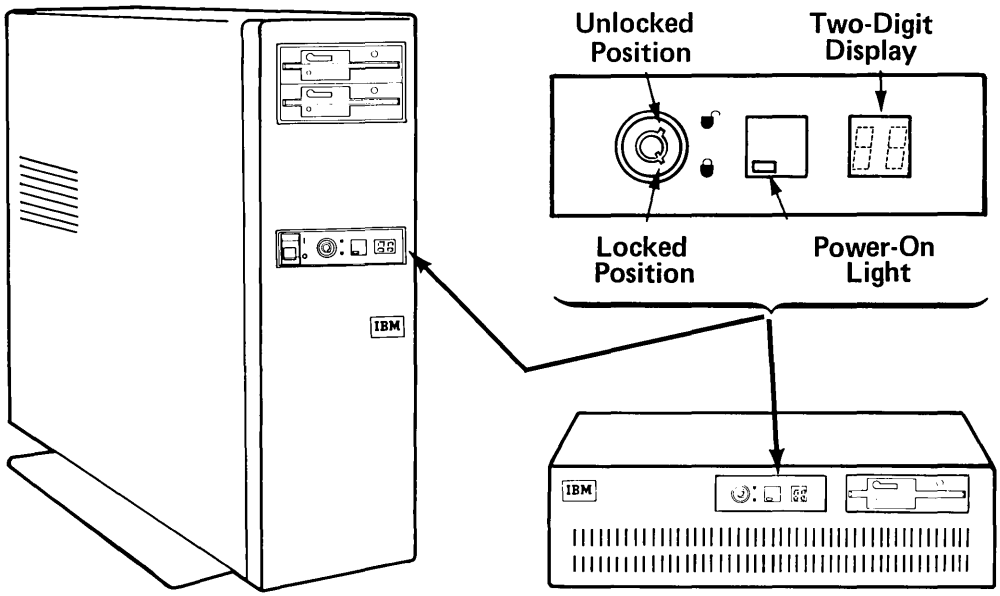


Figure 9-20. Operator Panel

Two-Digit Display

The two-digit display on the front of the system unit indicates the operation of the internal POSTs that are run each time the unit is turned on. The digits indicate 88 (all sections on) when the power is first turned on. During POSTs, a sometimes very rapid sequence of numbers appear, indicating that corresponding numbered tests are being run. Although the numbers may sequence too fast to read, machine operation stops when an error requiring repair action is detected, and the number of the test that detected the error is displayed. If the system cannot complete its POST or the loading of its software, the indicators display a number denoting the presence of an error. The indicators will be blank if the program has loaded correctly.

Keylock

The keylock deactivates the keyboard and locks the cover on. You can lock the system whether the power is on or off. Turn the key clockwise to lock the system or counterclockwise to unlock it. The key can be removed in either position.

When the keylock is in an unlocked position, you can:

- Operate the system
- Remove the frame cover on a IBM 6150.
- Remove the frame cover on a IBM 6151 after removing the cover thumbscrews.

When the keylock is in the locked position, you cannot:

- Operate the keyboard and optional locator
- Remove the covers without physical damage to the hardware.

IBM 6150 Battery Connector

On the IBM 6150 System Unit only the battery plugs onto a connector mounted, on the back, near the center of the operator panel printed circuit board back.

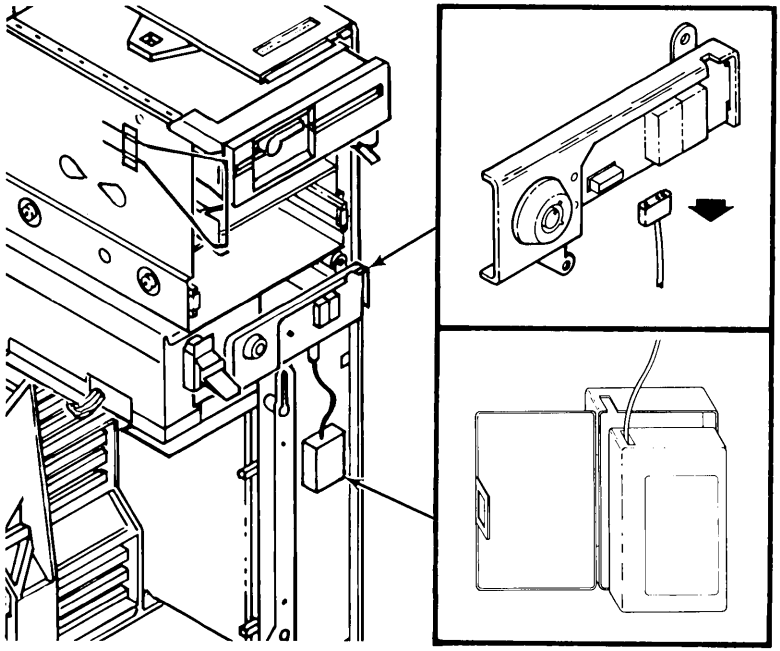


Figure 9-21. IBM 6150 Battery Connector

Section 10. Power

CONTENTS

About this Section

This section contains the system unit power specifications for the RT PC system. Included are the input and output power specifications and power signal definitions.

Input Power

The following tables lists the input voltages and amperes for the IBM 6150 and IBM 6151.

Volts (AC)	Frequency (Hz)	Current (Amperes)
Low range 90-137 Vac	50/60 Hz	Maximum 10 amperes (Includes 1.4 amps for ac outlet)
High range 180-259 Vac	50/60 Hz	Maximum 6 amperes (Includes 0.8 amps for ac outlet)

Figure 10-1. IBM 6151 Input Power

Note: The maximum in-rush current is 110 amperes.

Volts (AC)	Frequency (Hz)	Current (Amperes)
Low range 90-137 Vac	50/60 Hz	Maximum 10 amperes (Includes 1.4 amps for ac outlet)
High range 180-259 Vac	50/60 Hz	Maximum 6 amperes (Includes 0.8 amps for ac outlet)

Figure 10-2. IBM 6150 Input Power

Note: The maximum in-rush current is 110 amperes.

Output Power

The power supply provides +5, -5, +12, and -12Vdc. The following figures show the output connector pin assignments and the regulation tolerance for the voltages.

Note: The power supply also supplies high or low range Vac for the displays.

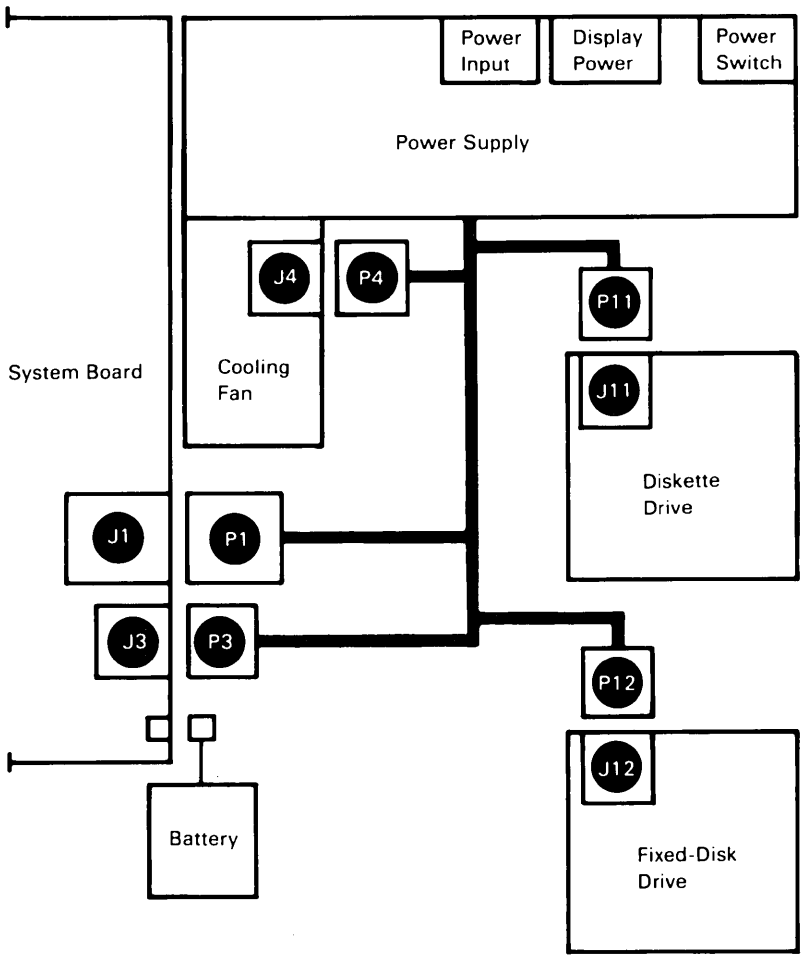
Conn	Pos	Voltage	Conn	Pos	Voltage
P11	1	+12 Vdc	P1	14	Not used
P11	2	Ground	P1	15	Not used
P11	3	Ground	P1	16	Not used
P11	4	+5 Vdc	P1	17	Ground
			P1	18	Ground
P12	1	+12 Vdc	P1	19	Ground
P12	2	Ground	P1	20	Ground
P12	3	Ground	P1	21	Ground
P12	4	+5 Vdc	P1	22	Ground
			P1	23	Ground
P1	1	+5 Vdc	P1	24	Ground
P1	2	+5 Vdc			
P1	3	+5 Vdc	P4	1	+12 Vdc
P1	4	+5 Vdc	P4	3	Ground
P1	5	+5 Vdc			
P1	6	+5 Vdc	P3	1	Sig. Gnd.
P1	7	+5 Vdc	P3	2	Key plug
P1	8	+5 Vdc	P3	3	Not used

Figure 10-3 (Part 1 of 2). IBM 6151 Power Connector Outputs

Conn	Pos	Voltage	Conn	Pos	Voltage
P1	9	+12 Vdc	P3	4	POR
P1	10	-5 Vdc	P3	5	EPOW
P1	11	Not used	P3	6	+5 Vdc
P1	12	Not used	P3	7	Not used
P1	13	-12 Vdc	P3	8	Not used

Figure 10-3 (Part 2 of 2). IBM 6151 Power Connector Outputs

IBM 6151 Power Distribution Diagram



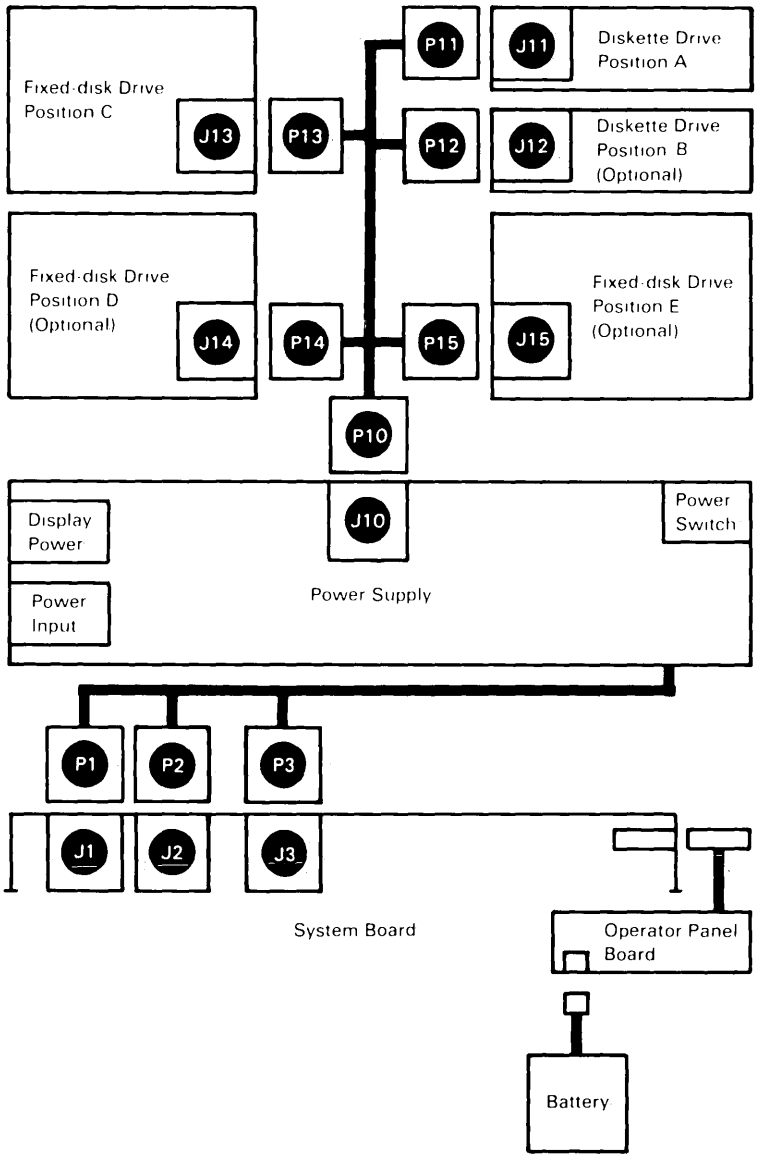
Nominal Output	Load Current Minimum (A)	Load Current Maximum (A)	Regulation Tolerance	Connector
+5 Vdc	12.00	37.00	+5% to -2%	P1, P3
+5	12.00		+5% to -3%	P11, P12
-5	0.01	0.30	+10% to -8%	P1
+12	1.50	7.20	+5% to -4%	P1, P4
+12	1.50		+5% to -4%	P11, P12
-12	0.01	0.60	+10% to -8%	P1

Figure 10-4. IBM 6151 Power Output Capability

Conn	Pos	Voltage	Conn	Pos	Voltage
P1	1	+5 Vdc	J10	1	+5 Vdc
P1	2	+5 Vdc	J10	2	+5 Vdc
P1	3	+5 Vdc	J10	3	+12 Vdc
P1	4	+5 Vdc	J10	4	+12 Vdc
P1	5	Ground	J10	5	Ground
P1	6	Ground	J10	6	Ground
P1	7	Ground	J10	7	Ground
P1	8	Ground	J10	8	Ground
			J10	9	Ground
P2	1	+5 Vdc			
P2	2	+5 Vdc	P3	1	EPOW
P2	3	+5 Vdc	P3	2	Key plug
P2	4	+5 Vdc	P3	3	POR
P2	5	Ground	P3	4	-12 Vdc
P2	6	Ground	P3	5	+12 Vdc
P2	7	Ground	P3	6	-5 Vdc
P2	8	Ground	P3	7	Ground
			P3	8	Ground
			P3	9	Ground

Figure 10-5. IBM 6150 Power Connector Outputs

IBM 6150 Power Distribution Diagram



Nominal Output	Load Current Minimum (A)	Load Current Maximum (A)	Regulation Tolerance	Connector
+5 Vdc	12.00	45.00	+5% to -2%	P1, P2
+5	12.00		+5% to -3%	J10
-5 Vdc	0.01	0.60	+10% to -8%	P3
+12	1.50	9.60 *	+5% to -4%	P3, J10
-12	0.01	0.80	+10% to -8%	P3

Figure 10-6. IBM 6150 Power Output Capability

Note: * Start current 16.5A

Power Supply Signals

Power On Reset

The power supply will provide an active low 'power supply reset' signal for a minimum of 200 milliseconds and a maximum of 500 milliseconds after the +5V has reached its operational level.

Early Power Off Warning

The 'early power off warning' (EPOW) signal is activated by the power supply if an input ac fault condition is detected. The EPOW guarantees a minimum of 2.0 milliseconds of in tolerance dc supply voltages.

Power Supply Protection

Output protection

The power supply provides overvoltage, undervoltage and overcurrent protection monitored at each voltage level in the power supply. An overcurrent condition will not damage the power supply. Protect circuits are electronic devices that are reset by a power-off or power-on cycle.

No load operation

The power supply will operate with no load applied to the outputs and be within 30% of nominal dc voltages.

Thermal protection

The system will power off if operational temperature limits are exceeded.

Output voltage sequencing	No special sequencing is provided, however, under normal conditions all output voltages will reach regulated levels within 500 milliseconds. The negative voltage levels will equal or lead the positive levels when power is turned on and lag or equal the positive levels when power is turned off.

Section 11. System Processor and MMU

CONTENTS

About this Section

This section contains information about the system processor and the memory management unit. It also provides information for the system instruction set and the instruction formats.

System Processor

System Organization And Control

A RT PC system consists functionally of a system processor, a memory management unit, system memory, a I/O channel converter, and input/output adapters. This structure is shown below:

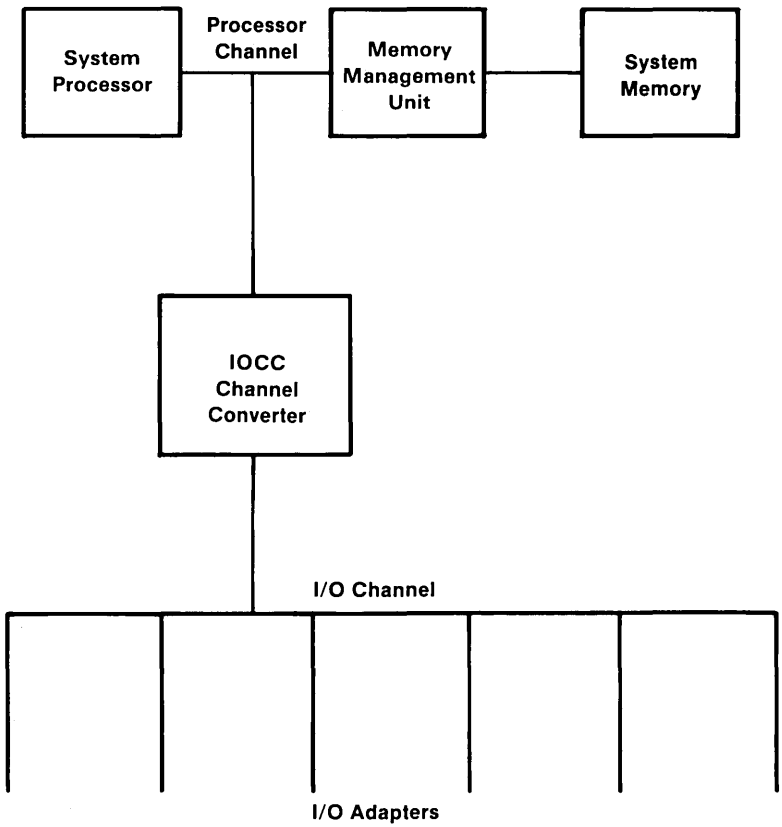


Figure 11-1. System Processor Block Diagram

System Memory

The system processor provides direct addressable system memory for data and instructions. The data units in system memory are shown below.

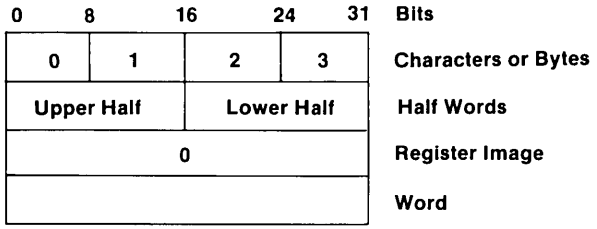


Figure 11-2. Data Units in System Memory

System memory is organized as a sequence of 32-bit words, each consisting of four 8-bit bytes. Bytes in the system memory are consecutively numbered, left to right, starting with 0. Each number is considered the address of the corresponding byte. All addresses are computed as byte addresses. The address of a word has zeros in the two low-order bits. The address of a halfword has one 0 in the low-order bit. Instructions must be located on halfword boundaries.

All memory effective addresses (base address plus displacement) are computed as 32-bit quantities. Wrap around is allowed and occurs on a 32-bit basis, that is, system memory addressing wraps around from the maximum byte address of 4,294,967,295 to address 0.

Processor Channel

The processor channel provides a 32-bit plus 4 parity-bits wide synchronous, which cycles at twice the processor cycle rate. The purpose of the channel is to enhance processor performance by having a channel dedicated to processor and memory transfers. All data and addresses transferred on the channel are multiplexed on the 32 address data lines. An optional address extension channel consisting of 8 address lines plus parity is provided for systems using 32-bit addressing. Devices connected to the processor channel are identified by a 5-bit tag identifier that devices send on the processor channel simultaneously with their request. Arbitration is accomplished by using a linear arbitration mechanism with devices connected in a daisy chain. The system processor, memory management unit, and the I/O channel converter (IOCC) interface logic are the only system components attached to this processor channel.

Programmed I/O

Programmed I/O is allowed using the Input/Output Read (IOR) and Input/Output Write (IOW) instructions. With these instructions, I/O devices can be read or written synchronously with program execution.

Processor

The processor contains the sequencing and processing controls for instruction execution, interrupt action, the system timer, and other control related functions.

Instructions are grouped into ten classes:

- Memory access
- Computation
- Branching
- Traps
- Moves and inserts
- Arithmetic operations
- Shifts
- System control
- Input/output.

A separate subsection is devoted to each instruction class (see "Instruction Set" on page 11-27).

Processor States

Four states of the processor are defined:

- Execution state
- Wait state
- Check stop state
- Stopped state.

Execution State

The processor is in the execution state when it is executing instructions. In the executing state, instruction fetching and execution proceeds in the specified manner. Interrupts may occur between instructions as specified in "Instruction Set" on page 11-27.

Wait State

After the processor executes the Wait instruction, it is in the wait state. No other instructions are fetched or executed while the processor is in the wait state. The processor leaves the wait state and enters the executing state when an interrupt for which the processor is enabled occurs (see "Interrupts" on page 11-19). The instruction address in the old program status for the priority level associated with the interrupt contains the address of the instruction immediately following the Wait instruction. The I/O pin 'wait' is active when the processor is in the wait state.

Check Stop State

When the processor is in the check stop state, instructions are not executed, interrupts do not occur, and system interface operations may be suspended. The processor enters the check stop state when one of the following occurs:

1. An error is detected during power-on diagnostics.
2. A machine check error is detected and the check stop mask is 0.
3. A program check or machine check error is detected and the processor is servicing a machine check error.
4. A program check error is detected and the processor is servicing a program check error.

The check stop condition is cleared during a power-on reset. The processor machine check is described in "Machine-Check Status" on page 11-86 and the program check is described in "Program-Check Status" on page 11-87

Stopped State

The stopped state is entered as a result of operator action from a control panel. Operator-initiated load, display, and stepping functions occur in the stopped state.

Privileged and Unprivileged States

The selection between privileged and unprivileged state determines whether the full set of instructions is valid. In privileged state, all instructions are valid. In unprivileged state, only instructions that cannot be used to affect system integrity are valid. Privileged instructions include those that inspect or modify any system control registers (except the condition status or multiplier quotient registers), the Load Program Status instruction, and the Wait instruction. A privileged instruction encountered in the unprivileged state constitutes a privileged instruction exception and causes a program check. Refer to "Privileged Instructions" on page A-8 for a list of the privileged instructions.

The processor is in unprivileged state when bit 21 of the interrupt control status (ICS) register is a one. The processor is in privileged state when bit 21 of the ICS is a zero.

General-Purpose Registers (GPR)

The processor provides sixteen 32-bit general-purpose registers (GPRs). All manipulation of data is performed in the GPRs.

Each GPR consists of an upper and lower half of 16 bits each. The GPR may be partitioned into four 8-bit characters, C0, C1, C2, and C3. The general-purpose register organization is shown in Figure 11-3.

For computation purposes, the content of a GPR is treated as either a signed algebraic quantity, an unsigned positive quantity, or an unstructured logical quantity. In a GPR, an algebraic quantity is represented by 32 bits in twos complement form.

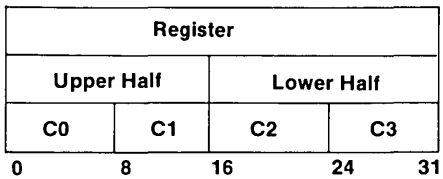


Figure 11-3. General-Purpose Registers

GPR 0 GPR 1	Register Pair
GPR 2 GPR 3	Register Pair
GPR 4 GPR 5	Register Pair
GPR 6 GPR 7	Register Pair
GPR 8 GPR 9	Register Pair
GPR 10 GPR 11	Register Pair
GPR 12 GPR 13	Register Pair
GPR 14 GPR 15	Register Pair

Figure 11-4. General Purpose Register Pairs

System Control Registers (SCR)

Sixteen 32-bit SCRs exist in the processor. An entire SCR or fields within an SCR are assigned to particular facilities in the system such as interrupt, processor, and system timer. The register organization for SCRs is shown below. Some SCRs and SCR fields are reserved and are not assigned to any system facility. The source bits are ignored on an attempt to set the reserved bits of an SCR. When the reserved bits of an SCR are fetched, the resulting values are unpredictable.

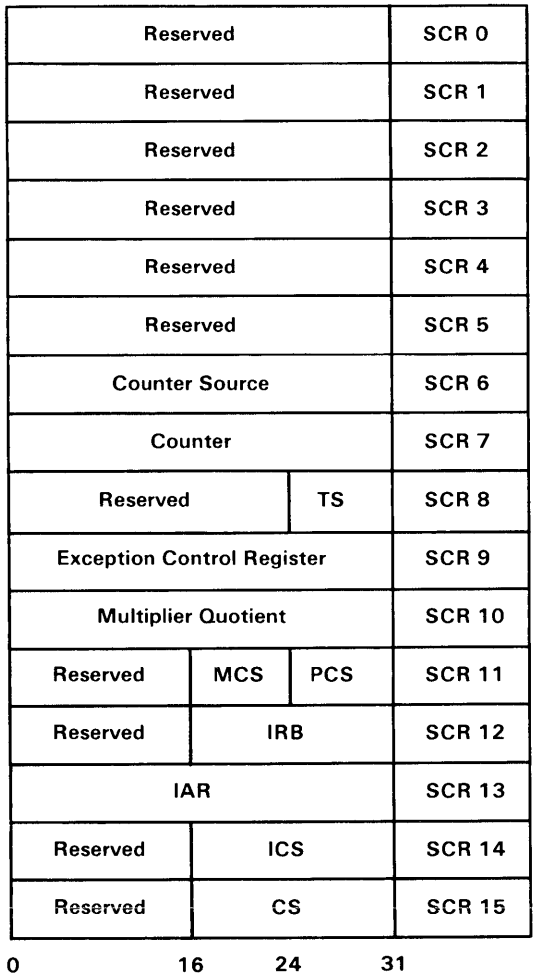


Figure 11-5. Organization

The system control registers are defined as follows:

SCR **Description**

0 - 5 **Reserved**

6 **Counter Source:** Is a 32-bit value that is automatically loaded into the counter when an alarm occurs.

7 **Counter:** Is a 32-bit count-down counter. The counter is decremented from an external source connected to the I/O pin '-Timer Clock'. The counter is updated on an inactive to active transition of '-Timer Clock'. Execution of processor instructions is suspended during the counter update. When the counter is decreased from 1 to 0, the value contained in the counter source is loaded into the counter and the alarm action is initiated. The alarm action sets a bit in the interrupt request buffer (IRB) whose priority level corresponds to the timer interrupt priority in SCR 8, if the timer is enabled. The alarm also sets the interrupt status bit to 1, and updates the timer status. The contents of the counter source are not altered.

8 **Timer Status (TS):** Bits 24-31 of SCR 8, is defined as follows:

Bit 24	Reserved
Bit 25	Enable: When 0, no interrupts are created. This does not start or stop the counter, but enables or disables the setting of IRB bits. At power-on reset, this bit is 0.
Bit 26	Interrupt status: When 1, an alarm has occurred. This is set only if an alarm has occurred, and the enable bit is set to 1. This bit is reset by software when the counter is serviced. Software can reset this bit by executing a Clear SCR Bit (CLRSB) instruction.
Bit 27	Overflow: When 1, more than one alarm has occurred before the interrupt status bit has been reset. This bit is also reset by software when the counter is serviced.
Bit 28	Reserved
Bits 29-31	Timer interrupt priority: A timer alarm sets an IRB bit, corresponding to the priority level specified by this field, if the timer is enabled.

9 **Exception Control Register:** See Appendix B for definition.

10 **Multiplier Quotient:** The MQ provides a GPR extension to accommodate the product for the Multiply Step instruction and the dividend for the Divide Step instruction. (See "Instruction Set" on page 11-27).

11 **Machine Check Status and Program Check Status:** Bits 16-23 of SCR 11 are called the machine check status (MCS) and bits 24-31 are called the program check status (PCS). These two fields are used for error identification and are described in detail in "Machine-Check Status" on page 11-86 and "Program-Check Status" on page 11-87.

12 **Interrupt Request Buffer:** Bits 16-31 of SCR 12 are called the interrupt request buffer (IRB). The interrupt request buffer allows interrupt requests to be generated under program control. For more detailed information refer to "Interrupt Request Buffer" on page 11-24.

13 **Instruction Address Register:** The instruction address register is a 32-bit register that contains the address of the next instruction to be executed. Since all instructions lie on halfword boundaries, the low-order bit (bit 31) of the instruction address register is 0. Accesses for instructions may require the fetching of a word, a halfword, or the low-order halfword of a word followed by the high-order halfword of the next consecutive word in system memory.

When executing an instruction, the content of the instruction address register is increased by the length of the current instruction. Should this instruction be a successful branch or jump instruction, the content of the instruction address register is changed to the address of the branch or jump target instruction. The IAR contains the address of the next instruction when the IAR is saved as part of the program status and when a system control instruction to read the IAR is executed.

14 **Interrupt Control Status:** Bits 16-31 of SCR 14 are called the interrupt control status (ICS). The ICS contains the following:

- parity error retry interrupt enable bit
- memory protect bit
- unprivileged state bit,
- translate mode bit
- interrupt mask
- check stop mask
- register set number
- processor priority.

The ICS is described in "Interrupt Control Status" on page 11-24.

15 **Condition Status:** Bits 16-31 of SCR 15 are called the condition status (CS). The condition status contains information about the results of certain operations and provides a mechanism for decision making. The condition status is defined as follows:

Bit(s)	Name and Definition
16-23	**Reserved**
24	**Permanent zero:** Set to 0 whenever the condition status is loaded, and cannot be set to 1. Its presence provides for a guaranteed branch or jump by use of a branch on the not-condition bit or jump on the not-condition bit instruction specifying the permanent zero bit.

25	**Less Than (LT)**: Set to 1 during logical, shift, and certain arithmetic instructions if the result is negative; otherwise, it is set to 1. Less than is also set during compare instructions to indicate the relative algebraic magnitudes of the comparands.
26	**Equal (EQ)**: Set to 1 during logical, shift, and certain arithmetic instructions if all bits of the result are zeros; otherwise, it is set to 0. It is also set during compare instructions if the comparands are equal.
27	**Greater Than (GT)**: Set to 1 during logical, shift, and certain arithmetic instructions if the sign bit of the result is 0 and the result is nonzero; otherwise, it is set to 0. It is also set during compare instructions to indicate the true relative algebraic magnitudes of the comparands.
28	**Carry Zero (C0)**: Set to 1 during certain arithmetic instructions if the operation generates a carry out of bit position zero; otherwise, it is set to 0.
29	**Reserved**
30	**Overflow (OV)**: Set to 1 during certain arithmetic instructions if the signed result of the operation cannot be represented in 32 bits; otherwise, it is set to 0.
31	**Test Bit (TB)**: Set by the move-to-test bit instructions, where a specified bit of half of a register is moved to the test bit. It is also affected by instructions that load or directly alter the Condition Status register.

All bits of the Condition Status, except the Permanent Zero bit, can be set through use of the move to SCR instructions.

A 4-bit field in the conditional branch instructions specifies the condition status bit to be tested. A 0 in the 4-bit field of a branch instruction specifies bit 16 of the condition status, a 1 specifies bit 17 of the condition status, and so on. A 3-bit field in the conditional jump instructions specifies the condition status bit to be tested. A 0 in this 3-bit field specifies bit 24, a 1 specifies bit 25, and so on.

System Timer Facility

Many applications require a knowledge of real time for such functions as system counting, time slicing, time stamping, interval timing, and timing the productivity of operations. The system timer facility provides these functions.

For some devices, the device requirements may be such that additional timers are needed in the adapter. A more sophisticated timer can be provided by the I/O device, if needed.

The system timer frequency is programmable by system software. The default frequency is approximately 1 KHz. It is called the 1 millisecond timer clock and is provided by the real time clock module on the system board.

Counter (COU)

SCR 7 is called the COU and is a 32-bit count-down counter. The counter is decreased from an external 1 msec signal connected to the I/O pin '-Timer Clock'. The processor instruction execution is suspended during the counter update. When the counter is decreased from 1 to 0, the value contained in the counter source is loaded into the counter and the alarm action is initiated. Normal operations of this action continues by the time the next count pulse arrives. The alarm action sets a bit in the IRB whose priority level corresponds to the timer interrupt priority in SCR 8, if the timer is enabled. The alarm also sets the interrupt status bit to 1, and updates the timer status. The contents of the counter source are not altered.

Timer status

The timer status (TS), bits 24-31 of SCR 8, is defined as follows:

Bit 24	Reserved
Bit 25	Enable: When 0, no interrupts are created. This does not start or stop the counter, but enables or disables the setting of IRB bits. At power-on reset, this bit is 0.
Bit 26	Interrupt status: When 1, an alarm has occurred. This is set only if an alarm has occurred, and the enable bit is set to 1. This bit is reset by software when the counter is serviced. Software can reset this bit by executing a Clear SCR Bit (CLRSB) instruction.
Bit 27	Overflow: When 1, more than one alarm has occurred before the interrupt status bit has been reset. This bit is also reset by software when the counter is serviced.
Bit 28	Reserved
Bits 29-31	Timer interrupt priority: A timer alarm causes the setting of an IRB bit corresponding to the priority level specified by this field if the timer is enabled.

System Timer Operation

To provide an interval timer, the counter is directly loaded with a value corresponding to the amount of time until the interval is to expire.

To provide a fixed interval interrupt, an appropriate value is loaded into the counter source and is not changed. For example, if the counter source was loaded with X'05', and a 1 millisecond timer clock, the processor is interrupted every 5 milliseconds; if it was loaded with 250 (X'FA'), the processor is interrupted every one-fourth of a second. The software then updates internal memory locations and provides time-of-day in whatever format desired.

Loading the counter source does not alter the value in the counter. As a result, the interrupt interval corresponding to the value loaded into the counter source begins when the counter is decreased from 1 to 0, and the new counter source is loaded into the counter. To synchronize the counter with a new counter source, both the counter source and the counter must be loaded with the new counter source value.

Multiple simultaneous timings can be handled using the system timer as a resource. The counter is loaded from a queue whose entries are calculated to be the time from the completion of the previous entry until the time for the entry in question to be completed.

The value loaded into the timer interrupt priority (TS, bits 29-31) must be greater than or equal to zero (000) and less than or equal to six (110). These are the only values for which a corresponding IRB exists. A value of seven (111) in the timer interrupt priority sets no bit in the IRB when a timer interrupt occurs.

Interrupts

The interrupt facility permits the processor to change its status at the request of some other system component or processor conditions established by the program. Interrupt processing consists of saving the current program status and establishing the program status for servicing the interrupt. Interrupts only occur on instruction boundaries, but some instruction sequences are not interruptible. A Load Program Status (LPS) instruction is provided for software to return from an interrupt. Execution of an LPS instruction restores the IAR, the CS, and the ICS to the values that existed when the interrupt occurred.

The processor may also change its status, because of error conditions, within the processor or a system component. Error processing consists of saving the current program status and establishing the program status for servicing the error. Errors are grouped into two classes: machine check errors and program check errors. These errors are discussed in detail in "Machine-Check Errors" on page 11-85 and "Program-Check Errors" on page 11-87.

The interrupt facility is a priority-based mechanism. This permits the servicing of higher priority functions to take precedence over the servicing of lower functions.

Interrupt sources consist of the seven external interrupt inputs (-REQI0-6), software interrupts posted through setting of bits in the IRB, and error conditions (either the '-TRAP' input, or internal errors) detected during system operation. The seven external interrupt inputs and software setting of IRB interrupt request bits are treated in the same manner by the system processor. The interrupt request level is compared to the current processor priority specified by bits in the ICS. If the interrupt request represents a higher priority than the current processor priority, and interrupts are enabled by the interrupt mask in the ICS, then the interrupt is taken.

Servicing an interrupt consists of saving the current processor status in the old PSW corresponding to the level of the interrupt request, and loading a new processor status from the new PSW corresponding to the level of the interrupt request. Saving the current processor status requires saving the address of the instruction, the condition status, and the ICS when the interrupt occurred.

Loading the new processor status requires loading the new IAR (containing the address of the interrupt service routine) and the new ICS from the new PSW. Saving the current processor status and loading the new processor status is performed automatically by system processor hardware. The GPRs are not automatically saved by hardware. Software is responsible for saving any GPRs modified by the interrupt service routine. Once the interrupt has been serviced, execution of the old program can be resumed by loading the old program status word through an LPS instruction. This will restore the IAR, the CS, and the ICS to the values that existed when the interrupt occurred.

In addition to the seven interrupt levels, the detection of error conditions can cause interrupts to the program check and machine check interrupt levels. Interrupts to the program check level consists of errors that are probably due to software errors (that is, detection of an invalid op-code, addressing error, detection of a privileged instruction exception,). Interrupts to the machine check level consists of errors that are probably due to hardware errors (that is, processor channel parity errors, and processor channel timeouts). In addition, an external input '-Trap' can be used by system components to cause a machine check interrupt. See "Machine-Check Status" on page 11-86 and "Program-Check Status" on page 11-87 for description of Machine Check and Program Check interrupts.

Processor Priority

Under normal system conditions, the processor executes instructions at a level of priority called the processor priority. The processor priority may assume one of eight levels as specified by a 3-bit field in the ICS. Priorities for the eight levels are represented by the following inequality: Priority of Level 0 > Priority of Level 1 >...> Priority of Level 7.

The processor priority may be changed either by an interrupt or by an instruction that modifies the processor priority. There are two sources of interrupts: an interrupt condition signaled through the interrupt request buffer, or an interrupt condition signaled by some system component through the seven interrupt request inputs '-REQ10-6'.

The processor may also execute instructions at two levels that are not accessible through the interrupt facility. These levels are provided for the reporting and servicing of machine check and program checking error conditions as discussed in "Machine-Check Errors" on page 11-85 and "Program-Check Errors" on page 11-87.

Interrupt Request Priority

Interrupt requests occur on one of seven priority levels. Priorities for the seven levels are represented by the following inequality: Priority of level 0 > Priority of Level 1 >...> Priority of Level 6.

The processor may execute instructions with a processor priority of 7, but interrupt requests with a priority of 7 cannot occur.

Interrupt Priority Assignment

A bit being set to 1 in the interrupt request buffer causes an interrupt request to the level corresponding to that bit. Timer interrupts cause an interrupt request (through the IRB) to the level specified in the timer status. A system component causes an interrupt request at a level determined by the attachment of its interrupt request line '-REQI0-6' to the processor.

Point of Interrupt

Interrupts only occur on instruction boundaries. Furthermore, interrupts are prevented from occurring within certain instruction sequences. A branch with execute instruction and its subject instruction cannot be interrupted.

A branch with execute and its subject instruction is considered to be a unit, and interrupts only occur before or after the unit is executed. See "Instruction Set" on page 11-27.

Error Handling

If the processor is executing an error routine as a result of a machine check or program check error condition, all interrupt requests from system components and interrupts signaled through the IRB remain pending.

Program Status

The program status consists of the contents of the following system control registers:

- Instruction address register
- Condition status
- Interrupt control status.

On interrupt, the current program status is automatically saved in the old program status location. The program status for servicing the interrupt is loaded from the new program status location, except for the condition status. The condition status is not changed by loading the new program status.

Old and New Program Status Pairs

An old and new program status pair consists of eight bytes of old program status and eight bytes of new program status. When an interrupt occurs, the old and new program status pair is specified by the priority level of the interrupt.

Location of Old and New Program Status Pairs

The program status save area in system memory contains the old and new program status pairs. Two old and new program status pairs are for machine check and program check error handling. One old and new program status pair is used for the Supervisor Call (SVC) instruction, and the remaining seven are for interrupt servicing. The SVC old and new program status pair contains a 16-bit SVC interrupt code which is generated by execution of the SVC instruction (See "Supervisor Call (SVC)" on page 11-82). The 16-byte old and new program status pairs are located in consecutive system memory locations. The program status save area is located at addresses X'100' through X'19F'. Address translation is disabled for storing of the old program status and loading of the new program status. The program status save area in system memory is shown below:

Memory Address	Definition
X'100'	Old and New PS Pair 0
X'110'	Old and New PS Pair 1
X'120'	Old and New PS Pair 2
X'130'	Old and New PS Pair 3
X'140'	Old and New PS Pair 4
X'150'	Old and New PS Pair 5
X'160'	Old and New PS Pair 6
X'170'	Machine Check Old and New PS Pair
X'180'	Program Check Old and New PS Pair
X'190'	SVC Old and New PS Pair

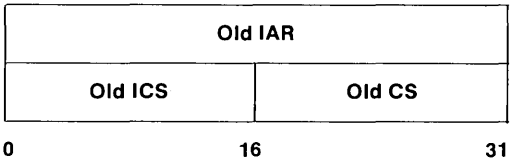


Figure 11-6. Old Program Status

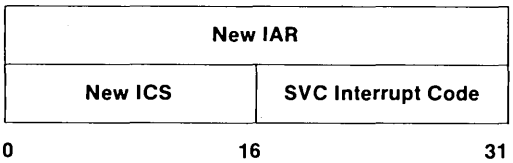


Figure 11-7. New Program Status

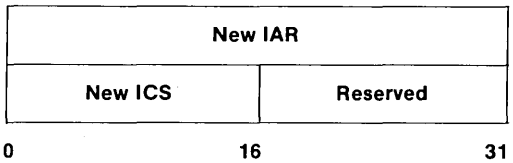


Figure 11-8. SVC New Program Status

IAR = Instruction Address Register
ICS = Interrupt Control Status
CS = Condition Status
SVC = Supervisor Call

Note: Reserved bits in the old program status are set to unpredictable values. Reserved bits in the new program status are ignored.

Interrupt Registers

Two fields are provided within the system control registers to support the interrupt facility. They are the interrupt request buffer and the interrupt control status.

Interrupt Request Buffer

The Interrupt Request Buffer (IRB) is a 16-bit field in SCR 12 and has the following format:

Bit(s)	Interrupt Request Level
16	Level 0
17	Level 1
18	Level 2
19	Level 3
20	Level 4
21	Level 5
22	Level 6
23-31	Reserved

The IRB can generate interrupt requests under software control. Setting an IRB bit to 1 causes an interrupt request to the level corresponding to that bit. The interrupt request remains active until the bit is cleared by software.

If bit 25 of SCR 8 is 1 (enabled), a timer alarm caused by the counter being decreased from 1 to 0 sets a bit in the IRB, which generates an interrupt request. The set bit in the IRB is determined by the timer interrupt priority in the timer status.

Interrupt Control Status

The interrupt control status (ICS) is a 16-bit field in SCR 14 with the following format:

Note: Bits are on (1) unless stated otherwise.

Bit(s)	Name and Description
16-17	**Reserved**
18	**Reserved**
19	**Parity Error Retry Interrupt Enable**: Enables interrupts when a processor channel retry successfully completes a processor-generated transfer that was previously unsuccessful due to detection of a parity error. A successful parity error retry interrupt will cause a level 0 interrupt by setting the interrupt request level 0 bit (bit

16) in the interrupt request buffer. The processor channel check bit (bit 16) in the machine check status indicates the cause of the interrupt.

20	**Memory Protect**: Enables address checking in the memory controller.
21	**Unprivileged State**: If bit 21 equals 0, the processor is in privileged state. If bit 21 equals 1, the processor is in unprivileged state.
22	**Translate Mode**: Enables address translation in the memory controller.
23	**Interrupt Mask**: Inhibits all system component, timer, and software interrupts on all levels. An inhibited interrupt remains pending.
24	**Check Stop Mask**: Prevents the processor from entering the check stop state on a machine check error.
25-27	**Register Set Number**: Specifies one of eight register sets as the active register set. The current system processor design implements one of the eight register sets. Bits 25-27 are ignored in this implementation.
28	**Reserved**
29-31	**Processor Priority**: Indicates the current processor priority level. Interrupt requests with priorities lower than or equal to the current processor priority are ignored.

Occurrence of Interrupts

An interrupt occurs when:

- A bit in the IRB being equal to 1 if the processor priority is lower than the priority corresponding to that bit of the IRB.
- The interrupt mask is 0.
- No system component is signaling an interrupt request on a higher level than that signaled through the IRB.

An interrupt occurs due to:

- A system component interrupt request if the processor priority is lower than the interrupt request.
- The interrupt mask is 0.
- The IRB is not signaling an interrupt request on a higher level than that signaled by the system component.

Interrupt Facility

The interrupt facility contains features that, if used improperly, may force the processor into an infinite hardware loop. When the processor loads the new program status for servicing an interrupt, it loads the processor priority from the ICS in the new program status location. The value in the processor priority in the new program status is completely under software control. This loaded processor priority must not be lower than the priority of the interrupt request which caused the interrupt. If the interrupt mask in the new program status is zero, the same interrupt request will immediately cause another interrupt. Multiple interrupts would continue to occur until a system component signals an interrupt request on a higher level, or until a power-on reset occurs.

Interrupt Servicing

The program should issue an IOW instruction to signal the device that the interrupt request is being serviced, and to reset the interrupt request bit in the device status.

1. The program should clear the interrupt request of the interrupting device as soon as possible after the point of interrupt. This allows the processor to determine the priority of the next highest interrupt request.
2. A Load Program Status (LPS) instruction is provided for software to return from an interrupt. The effective address of the LPS instruction points to the program status to which control is being returned. Normally control returns to the previously active program whose status is located in the old program status associated with the interrupt being serviced.

Instruction Set

Instructions are grouped into ten classes:

- Memory access,
- Address computation
- Branching
- Traps
- Moves and inserts
- Arithmetic operations
- Logical operations
- Shifts
- System control
- Input/output.

A separate section is devoted to each instruction class. Each instruction is specified in terms of mnemonic, operation code (op-code), length, and functional description.

Unassigned op-codes are reserved for future use. If these reserved op-codes are encountered by the processor, a program check error occurs. For more detailed information, see "Program-Check Errors" on page 11-87.

The system processor does not support dynamic instruction modification (self modifying code). Any attempt by software to modify an instruction may result in unpredictable operation.

The system processor provides a privileged state in which all instructions are valid, and an unprivileged state in which only instructions that cannot be used to affect system integrity are valid.

The following notation is used to describe each instruction:

GPR	General-purpose register (The word register is also used to denote a GPR.)
SCR	System control register
IAR	Instruction address register
IRB	Interrupt request buffer
MCS	Machine check status
CS	Condition status
ICS	Interrupt control status

R1,R2 or R3	These abbreviations denote fields in the instruction which specify one of the sixteen GPRs.
SRB	This denotes a field in the instruction which specifies an SCR.
I	This denotes a field of immediate data in the action.
N	This denotes a condition status bit number
JI	This denotes an 8-bit relative branch displacement in the JI format instructions.
BI	This denotes a 20-bit relative branch displacement in the BI format instructions.
BA	This denotes a 24-bit absolute branch address.
0/(R3)	This indicates the value 0 if R3 is specified as 0, else the content of register R3. (That is, if the R3 field is specified as 0, a value of 0 is used for the computation. If the R3 field is not 0, the content of the specified register is used for the computation. Register 0 cannot be used as the R3 register.
0[n]	This indicates a field of zeroes, n-bits wide.
//	Two parallel bars are used to indicate a concatenation of the two fields specified on either side of the bars.
(R3)	A register specification enclosed in parentheses indicates the content of the specified register.

The seven instruction formats (JI, X, D-Short, R, BI, BA and D) are shown on the following pages. Instructions are either 2 or 4 bytes in length. The first 4-, 5- or 8-bits of an instruction are called the operation code (op-code). The JI format has a 5-bit op-code. The X and D-Short formats both have 4-bit op codes. The R, BI, BA and D formats all have 8-bit op codes. Instructions of formats JI, X, D-Short and R are all 2 bytes long. Instructions of formats BI, BA and D are all 4 bytes long.

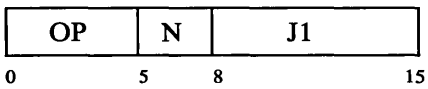
The R1, R2 and R3 fields specify GPRs. The SRB field specifies an SCR. The I field specifies a displacement of a memory address or an immediate value. The N field specifies a condition status bit. Relative branch displacements JI and BI are both signed binary numbers in twos complement form, while BA designates an absolute branch address.

Some R format instructions have an SRB, I, or N field instead of an R2 or R3 field.

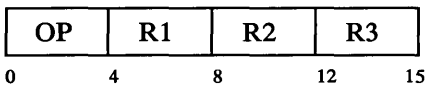
Instruction Formats

The seven instruction formats are shown below.

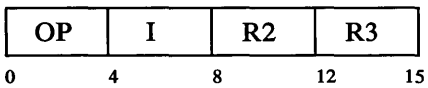
J1 Format



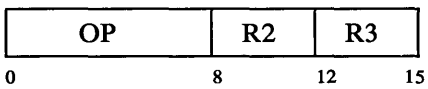
X Format



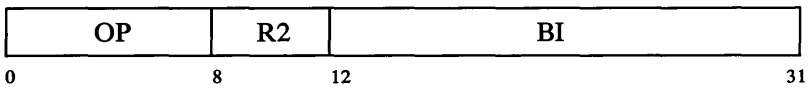
D-Short Format



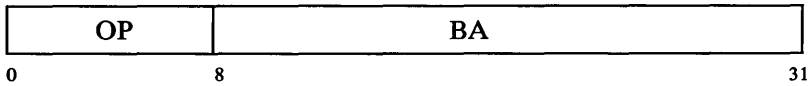
R Format



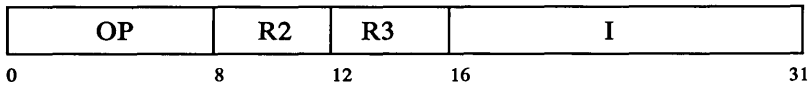
BI Format



BA Format



D Format



Memory Access

System memory is organized as a sequence of 8-bit bytes with a maximum addressing capability of 4,294,967,296 bytes. All memory effective addresses (base address plus displacement) are computed as 32-bit quantities. Wrap around is allowed and occurs on a 32-bit basis, (that is, system memory addressing wraps around from the maximum byte address of 4,294,967,295 to address 0).

All memory accesses are for a byte or multiples thereof. Instructions are provided to load or store a single character, a halfword, or a word into a general- purpose register. Memory accesses for halfwords and words ignore the low-order bit or pair of bits, respectively, of the effective address. The address of a halfword or word in system memory is the address of its leftmost byte. The condition status is not changed by any of these instructions.

A memory access to an invalid memory location sets the data address exception bit in the program check status and results in a program check. Refer to "Program-Check Status" on page 11-87 for a description of the program check status.

All memory access instructions are unprivileged.

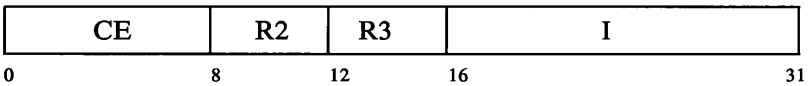
Load and Store Instructions

This group of instructions loads and stores data.

Load Character Short (LCS)

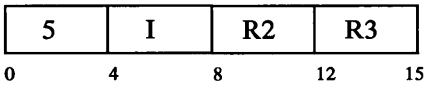
Characters C0 through C2 of register R2 are set to 0. Character C3 of register R2 is replaced by the character of memory addressed by 0/(R3) + 0[28]//I.

Load Character (LC)



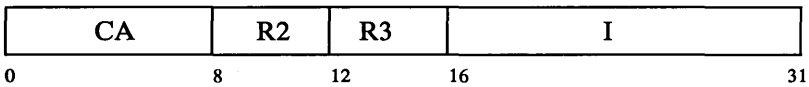
Characters C0 through C2 of register R2 are set to 0. Character C3 of register R2 is replaced by the character of memory addressed by 0/(R3) plus the sign-extended I-field.

Load Half Algebraic Short (LHAS)



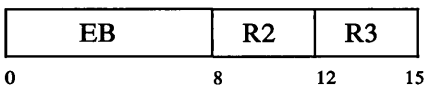
The lower half of register R2 is replaced by the halfword of memory addressed by 0/(R3)+0[27]//I//0[1]. The sign bit of the addressed halfword is extended through the upper half of register R2.

Load Half Algebraic (LHA)



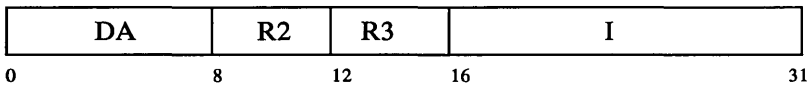
The lower half of register R2 is replaced by the halfword of memory addressed by 0/(R3) plus the sign-extended I-field. The sign bit of the addressed halfword is extended through the upper half of register R2.

Load Half Short (LHS)



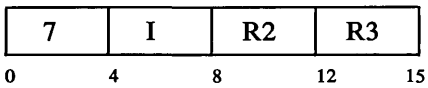
The upper half of register R2 is set to zero. The lower half of register R2 is replaced by the halfword of memory addressed by the content of register R3.

Load Half (LH)



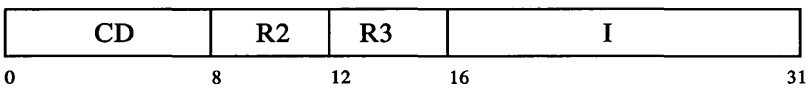
The upper half of register R2 is set to 0. The lower half of register R2 is replaced by the halfword of memory addressed by 0/(R3) plus the sign-extended I-field.

Load Short (LS)



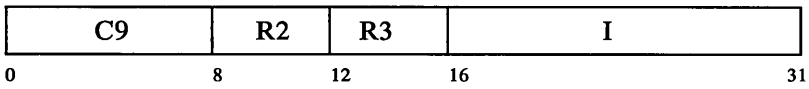
The content of register R2 is replaced by the word in memory addressed by 0/(R3) + 0[26]//I//0[2].

Load (L)



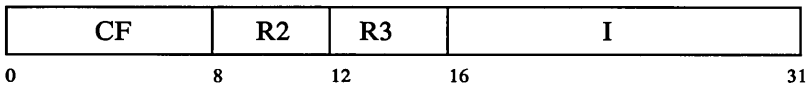
The content of register R2 is replaced by the word in memory addressed by 0/(R3) plus the sign-extended I-field.

Load Multiple (LM)



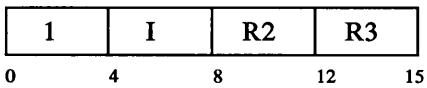
The content of registers R2 through 15 is replaced, respectively, by the consecutive words in memory beginning at the address given by 0/(R3) plus the sign-extended I-field.

Test and Set Half (TSH)



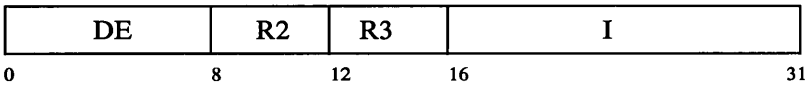
The upper half of register R2 is set to 0. The lower half of register R2 is replaced by the halfword of memory addressed by 0/(R3) plus the sign-extended I-field. Immediately following the read operation, the memory unit writes all ones in the high-order byte of the selected halfword without permitting any other memory operations between the read and the write. The low-order byte of the selected halfword is left unaltered.

Store Character Short (STCS)



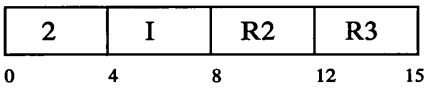
The character of memory addressed by 0/(R3) + 0[28]//I is replaced by character C3 of register R2.

Store Character (STC)



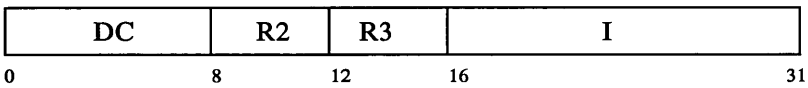
The character of memory addressed by 0/(R3) plus the sign extended I-field is replaced by character C3 of register R2.

Store Half Short (STHS)



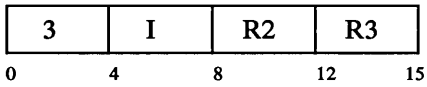
The halfword of memory addressed by 0/(R3) + 0[27]//I//0[1] is replaced by the lower half of register R2.

Store Half (STH)



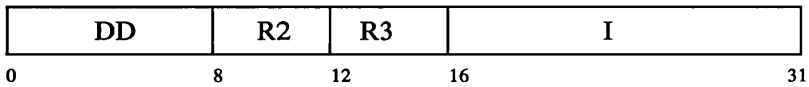
The halfword of memory addressed by 0/(R3) plus the sign-extended I-field is replaced by the lower half of register R2.

Store Short (STS)



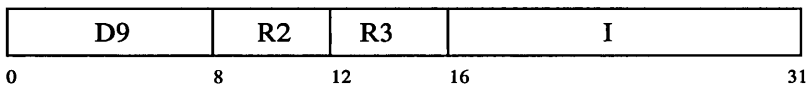
The word of memory addressed by 0/(R3) + 0[26]//I//0[2] is replaced by the content of register R2.

Store (ST)



The word in memory addressed by 0/(R3) plus the sign-extended I-field is replaced by the content of register R2.

Store Multiple (STM)



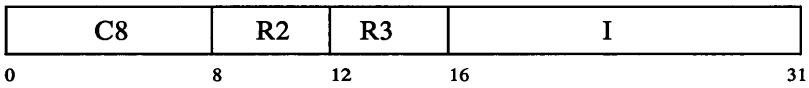
The consecutive words in memory beginning at the address given by 0/(R3) plus the sign-extended I-field are replaced, respectively, by the content of registers R2 through 15.

Address Computation

The address computation instructions operate only on the contents of the general-purpose registers. No memory references for operands occur. The resultant values are not inspected for address exceptions. The condition status is not changed by any of these instructions.

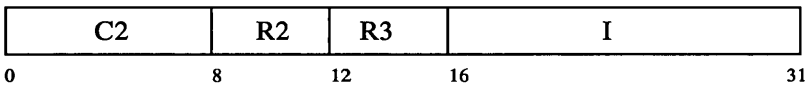
All address computation instructions are non-privileged.

Compute Address Lower Half (CAL)



The address specified by 0/(R3) plus the sign-extended I-field replaces the content of register R2.

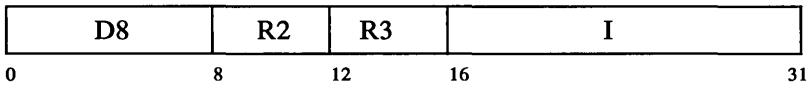
Compute Address Lower Half 16-Bit (CAL16)



The 16-bit address specified by 0/(R3)[16:31] + I replaces the content of register R2[16:31], and 0/(R3)[0:15] replaces the content of register R2[0:15].

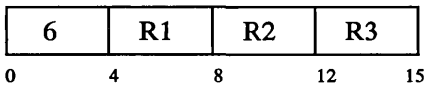
Note: This instruction is provided to assist in simulation of 16-bit architectures.

Compute Address Upper Half (CAU)



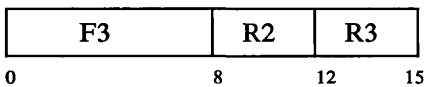
The address specified by 0/(R3) + I/ 0[16] replaces the content of register R2.

Compute Address Short (CAS)



The address specified by (R2) + 0/(R3) replaces the content of register R1.

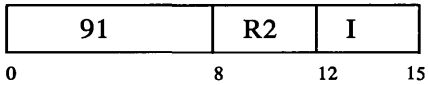
Compute Address 16-Bit (CA16)



The 16-bit address specified by (R2)[16:31] + (R3)[16:31] replaces the content of register R2[16:31], and (R3)[0:15] replaces the content of register R2[0:15].

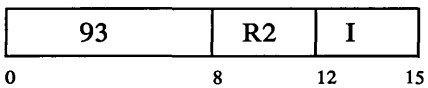
Note: This instruction is provided to help in simulation of 16-bit architectures.

Increment (INC)



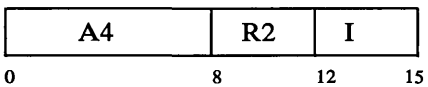
The field I, extended on the left with 28 zeroes, is added to the content of register R2, and the result is placed into register R2.

Decrement (DEC)



The field I, extended on the left with 28 zeroes, is subtracted from the content of register R2, and the result is placed into the register R2.

Load Immediate Short (LIS)



The content of register R2 is replaced by field I and extended on the left with 28 zeroes.

Branching

The normal sequential execution of instructions may be changed by using the branch instructions. These instructions permit subroutine linkage, decision making, and loop control and provide several different target addressing forms.

For every branch instruction, except jumps, there is a corresponding branch with execute instructions. The instruction, immediately following a branch with execute, is called the subject instruction, and is executed regardless of the branch decision, as if it preceded the branch. However, the subject instruction cannot affect the branch decision. Any condition status changes caused by the subject instruction occur after the branch decision has been made.

Subroutine linkage is provided by these branch and linkage instructions: BALA, BALAX, BALI, BALR, and BALRX. These instructions branch to a new instruction sequence but preserve a return address in an implicitly or explicitly designated general-purpose register. For the nonexecute forms of the instructions, the return address is the updated instruction address, that is the address of the halfword immediately following the branch and link instruction in memory. For the execute forms of the instructions, the return address is the address of the halfword that is 4 bytes beyond the end of the branch and link with execute instruction (it is the updated instruction address plus four). This allows 4 bytes following the branch and link with execute for the subject instruction. If the subject instruction requires only 2 bytes, the 2 remaining bytes are ignored.

Decision making and loop control are provided by these conditional branch and conditional jump instructions: BB, BBX, BBR, BBRX, BNB, BNBX, BNBR, BNBRX, JB, and JNB. For the conditional branch instructions, the branch decision is based on any specified state of any bit of the condition status. In these instructions, the value of N specifies the condition status bit with CS bit 16 specified by a value of 0, CS bit 17 by a value of 1, and so forth. For the conditional jump instructions the branch is based on any specified state of the rightmost 8 bits (bits 24-31) of the condition status. In this case, the value of the N-field of the jump instruction specifies the condition status bit with CS bit 24 specified by a value of 0, CS bit 25 by a value of 1, and so forth. For conditional branch instructions, the branch decision is based on any specified state of the rightmost 16 bits (bits 16-31) of the condition status. In this case, the value of the N field specifies which condition status bit is used for the branch decision, with CS bit 16 specified by a value of 0, CS bit 17 specified by a value of 1, and so forth. If a reserved bit in the condition status (bits 16-23) is specified, the branch decision is unpredictable.

The branch instructions provide three different branch target addressing forms: absolute, absolute immediate, and relative immediate. The instructions BALR, BALRX, BBR, BBRX, BNBR, and BNBRX are absolute instructions and specify, as an operand, a register that contains the 24-bit branch target address. The instructions BALA and BAL AX are absolute immediate instructions, where the full 24-bit branch target address is contained in the instruction. The instruction BALI, BA LIX, JB, BB, BBX, JNB, BNB, and BNBX are relative immediate instructions; each contains an immediate field that is sign extended, logically shifted left 1 bit, and added to the address of the branch instruction to calculate the branch target address. The jump instructions (JB and JNB) contain an 8-bit immediate field that allows a jump range of -127 to +128 halfwords from the jump

instruction. The branch (instructions BALI, BALIX, BB and BBX contain a 20-bit immediate field that allows a branch range of -524,286 to +524,289 halfwords from the branch instruction.

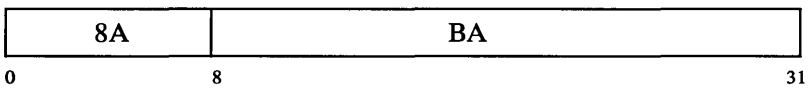
The branch with execute instruction and its subject instruction are considered to be a single instruction. Thus, interrupts are not honored between the execution of the branch with execute instruction and the execution of its subject instruction.

Certain instructions are not allowed to be the subject of a branch with execute instruction. Since the branch with execute instructions change the normal sequential execution of instructions, the subject instruction cannot be an instruction that also changes the instruction sequencing; otherwise the processor may be put in an unpredictable state. Thus, all branches, jumps, traps, Load Program Status, Supervisor Call, and Wait instructions cannot be subject instructions. The system processor provides hardware that detects these illegal instructions and causes a program check interrupt if any of these instructions occur at the subject of a branch with execute instruction.

Also, note that, in branch and link with execute instructions, the register containing the return address is available to the subject instruction. Therefore the subject instruction must be constructed so as not to modify the return address unintentionally. Finally, if the subject instruction is a Move From SCR, where the SCR is the IAR, (move from SCR 13) the results of the move are unpredictable.

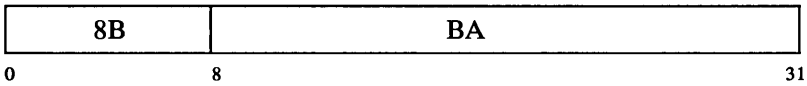
All branch and jump instructions are unprivileged.

Branch and Link Absolute (BALA)



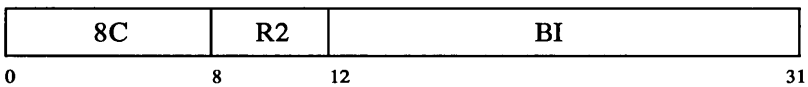
The content of register 15 is replaced by the updated instruction address, and the updated instruction address is replaced by the field BA, with its rightmost bit forced to 0.

Branch and Link Absolute with Execute (BALAX)



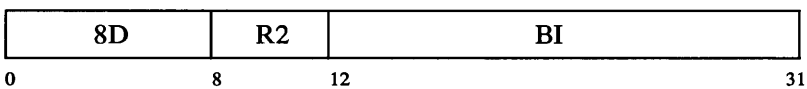
The content of register 15 is replaced by the updated instruction address incremented by 4, and the updated instruction address is replaced by the field BA with its rightmost bit forced to 0. The instruction immediately following the branch instruction is executed before the target instruction is executed.

Branch and Link Immediate (BALI)



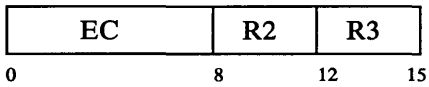
The content of register R2 is replaced by the updated instruction address. The updated instruction address is replaced by the field BI, sign extended, shifted left 1 bit, and added to the branch instruction address.

Branch and Link Immediate with Execute (BALIX)



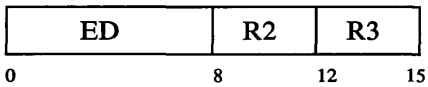
The content of register R2 is replaced by the updated instruction address incremented by 4. The updated instruction address is replaced by the field BI, sign extended, shifted left 1 bit, and added to the branch instruction address. The instruction immediately following the branch instruction is executed before the target instruction.

Branch and Link (BALR)



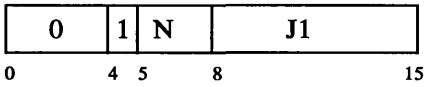
The content of register R2 is replaced by the updated instruction address. The updated instruction address is replaced by the content of register R3 with the rightmost bit forced to 0.

Branch and Link with Execute (BALRX)



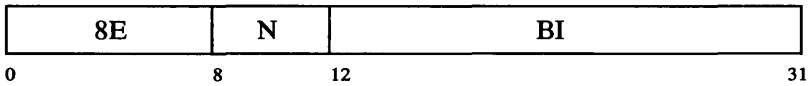
The content of register R2 is replaced by the updated instruction address incremented by 4, and the updated instruction address is replaced by the content of register R3 with the rightmost bit forced to 0. The instruction immediately following the branch instruction is executed before the target instruction.

Jump on Condition Bit (JB)



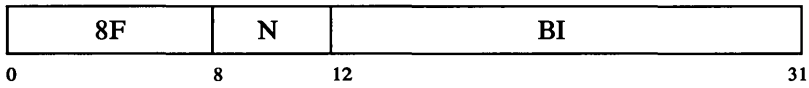
If the condition status bit specified by N is 1, the updated instruction address is replaced by the field JI, sign extended and shifted left one bit, added to the branch instruction address. If the specified condition status bit is 0, the updated instruction address is unaltered. The field N refers only to the rightmost 8 bits of the condition status (bits 24-31).

Branch on Condition Bit Immediate (BB)



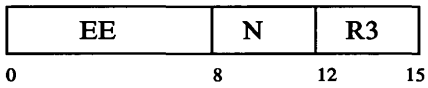
If the condition status bit specified by N is 1, the updated instruction address is replaced by the field BI, sign extended and shifted left one bit, added to the branch instruction address. If the specified condition status bit is 0, the updated instruction address is unaltered.

Branch on Condition Bit Immediate with Execute (BBX)



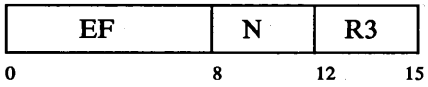
If the condition status bit specified by N is 1, the updated instruction address is replaced by the field BI, sign extended and shifted left one bit, added to the branch instruction address. The instruction immediately following the branch instruction is executed before the target instruction. If the specified condition status bit is 0, the updated instruction address is unaltered, and the subject instruction is executed in a normal manner.

Branch on Condition Bit (BBR)



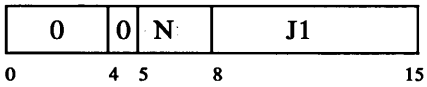
If the condition status bit specified by N is 1, the updated instruction address is replaced by the content of register R3 with the rightmost bit forced to 0. If the specified condition status bit is 0, the updated instruction address is unaltered.

Branch on Condition Bit with Execute (BBRX)



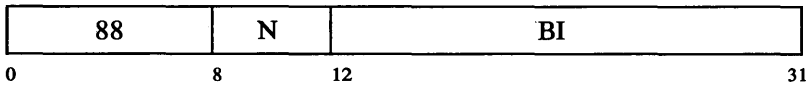
If the condition status bit specified by N is 1, the updated instruction address is replaced by the content of register R3 with the rightmost bit forced to 0. The instruction immediately following the branch instruction is executed before the target instruction. If the specified condition status bit is 0, the updated instruction address is unaltered, and the subject instruction is executed in a normal manner.

Jump on Not Condition Bit (JNB)



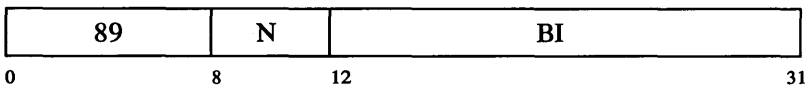
If the condition status bit specified by N is 0, the updated instruction address is replaced by the field JI, sign extended and shifted left 1 bit, and added to the branch instruction address. If the specified condition status bit is 1, the updated instruction address is unaltered. The field N references only the rightmost 8 bits of the condition status (bits 24-31).

Branch on Not Condition Bit Immediate (BNB)



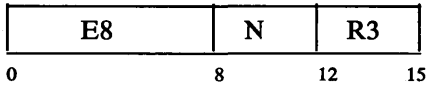
If the condition status bit specified by N is 0. The updated instruction address is replaced by the field BI, sign-extended and shifted left one bit, and added to the branch instruction address. If the specified condition status bit is 1, the updated instruction address is unaltered.

Branch on Not Condition Bit Immediate with Execute (BNBX)



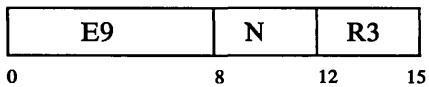
If the condition status bit specified by N is 0, the updated instruction address is replaced by the field BI, sign extended and shifted left one bit, and added to the branch instruction address. The instruction immediately following the branch instruction is executed before the target instruction. If the specified condition status bit is 1, the updated instruction address is unaltered, and the subject instruction is executed in a normal manner.

Branch on Not Condition Bit (BNBR)



If the condition status bit specified by N is 0, the updated instruction address is replaced by the content of register R3 with the rightmost bit forced to 0. If the specified condition status bit is 1, the updated instruction address is unaltered.

Branch on Not Condition Bit with Execute (BNBRX)



If the condition status bit specified by N is 0, the updated instruction address is replaced by the content of register R3 with the rightmost bit forced to 0. The instruction immediately following the branch instruction is executed before the target instruction. If the specified condition status bit is 1, the updated instruction address is unaltered, and the subject instruction is executed in a normal manner.

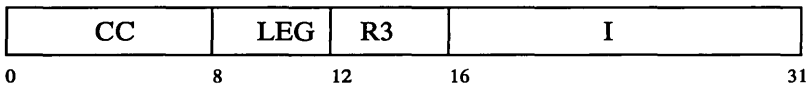
Traps

The trap instructions are provided to test for a specified set of conditions. If the conditions tested by a trap instruction are met, the program trap bit of the program check status is set to 1 and a program check occurs. If the tested conditions are not met, instruction execution continues with the next sequential instruction.

The comparisons are performed on operands that are treated as 32-bit unsigned integers (logical quantities). The condition status is not changed by any of these instructions.

All trap instructions are unprivileged.

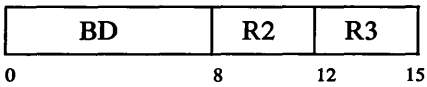
Trap On Condition Immediate (TI)



If any of the trap conditions specified by bits 9 through 11 are met by comparing the content of register R3 with the value of the sign extended I-field, the trap bit of the PC register is set, and a program check occurs. Trap conditions are selected by bits 9 through 11 as defined below.

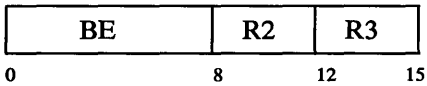
Bit	Definition
9	Trap if register R3 is less than the value of the sign extended I-field. The trap is enabled if the bit is 1 and disabled if 0.
10	Trap if register R3 is equal to the value of the sign extended I-field. The trap is enabled if the bit is 1 and disabled if 0.
11	Trap if register R3 is greater than the value of the sign extended I-field. The trap is enabled if the bit is 1 and disabled if 0.

Trap if Register Greater Than or Equal (TGTE)



If the content of register R2 is greater than or equal to the content of register R3, the trap bit of the PCS is set, and a program check occurs.

Trap if Register Less Than (TLT)



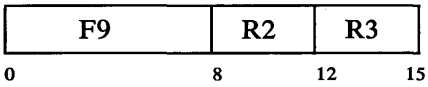
If the content of register R2 is less than the content of register R3, the trap bit of the PCS is set, and a program check occurs.

Moves and Inserts

This group of instructions concerns the movement of data between general-purpose registers, and between a general-purpose register and the test bit of the condition status. Except when data is moved into the test bit, none of these instructions alter condition status.

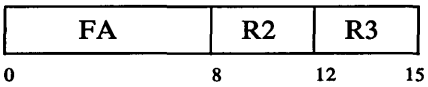
All move and insert instructions are unprivileged.

Move Character Zero From Three (MC03)



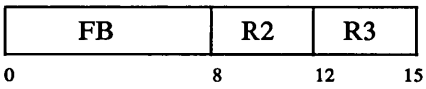
Character C0 of register R2 is replaced by character C3 of register R3.

Move Character One From Three (MC13)



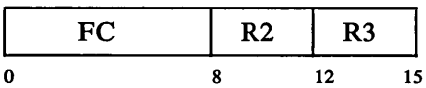
Character C1 if register R2 is replaced by character C3 of the register R3.

Move Character Two From Three (MC23)



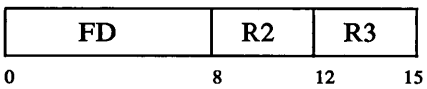
Character C2 of register R2 is replaced by character C3 of register R3.

Move Character Three From Three (MC33)



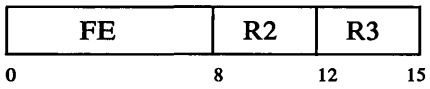
Character C3 of register R2 is replaced by character C3 of register R3.

Move Character Three From Zero (MC30)



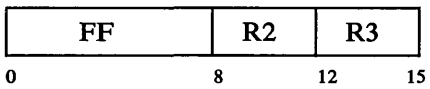
Character C3 of register R2 is replaced by character C0 of register R3.

Move Character Three From One (MC31)



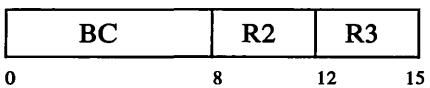
Character C3 of register R2 is replaced by character C1 of register R3.

Move Character Three From Two (MC32)



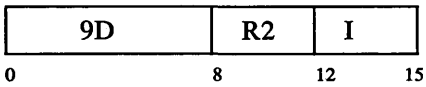
Character C3 of register R2 is replaced by character C2 of register R3.

Move From Test Bit (MFTB)



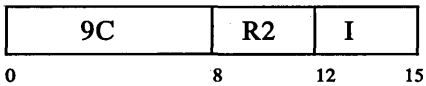
The bit of register R2 specified by the value of bits 27-31 of register R3 is set to the value of the condition status test bit.

Move From Test Bit Immediate Lower Half (MFTBIL)



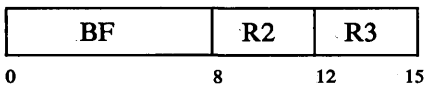
The bit of the lower half of register R2 specified by I is set to the value of the condition status test bit.

Move From Test Bit Immediate Upper Half (MFTBIU)



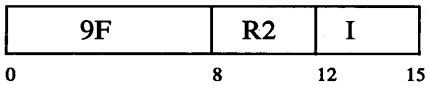
The bit of the upper half of register R2 specified by I is set to the value of the condition status test bit.

Move to Test Bit (MTTB)



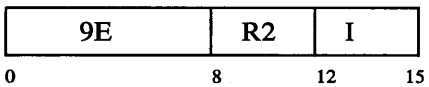
The condition status test bit is set to the value of the bit of register R2 specified by the value of bits 27-31 of register R3.

Move to Test Bit Immediate Lower Half (MTTBIL)



The condition status test bit is set to the value of the bit in the lower half of register R2 specified by I.

Move To Test Bit Immediate Upper Half (MTTBIU)



The condition status test bit is set to the value of the bit in the upper half of register R2 specified by I.

Arithmetic Operations

The arithmetic operations treat the general-purpose registers as 32-bit quantities in twos complement representation. Each of these instructions affects certain bits in the condition status field. However, the bits that are set and the manner in which they are set may vary according to the instruction that is executed.

The less than (LT) bit is set by all instructions except Multiply Step and Divide Step, and compares to indicate the sign of the result. That is, the LT bit is set to 1 if the sign bit of the result is 1. The arithmetic compare instructions set this bit to 1 if the algebraic magnitude of a given comparand is less than the algebraic magnitude of the other. The logical compare instructions set this bit to 1 if the unsigned magnitude of a given comparand is less than the unsigned magnitude of the other. The instructions Multiply Step and Divide Step do not affect this bit.

The equal (EQ) bit is set by all instructions except Multiply Step and Divide Step if the result is a field of 32 zeroes, or, in the case of the compare instructions, if the two comparands are equal. The Multiply Step and Divide Step instructions do not affect this bit.

The greater than (GT) bit is set by all instructions except Multiply Step and Divide Step, and compares to indicate the sign of a non-zero result; it is set to 1 if the sign bit of a non-zero result is 0. The arithmetic compare instructions set this bit if the algebraic magnitude of a given comparand is greater than the algebraic magnitude of the other. The logical compare instructions set this bit to 1 if the unsigned magnitude of a given comparand is greater than the unsigned magnitude of the other. The instructions Multiply Step and Divide Step do not affect this bit.

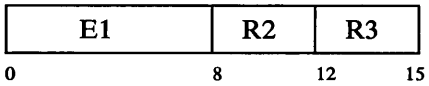
The carry of bit position zero (C0) bit in the condition status is set by all instructions except compares, Extend Sign, Divide Step, and Multiply Step to reflect the carry of bit position zero. The Extend Sign instruction does not affect C0, and the Multiply Step and Divide Step instructions set C0 according to certain multiply and divide conditions. Add operations set C0 to 1 if a carry occurs and to 0 if no carry occurs. Subtract operations set C0 to 0 if a borrow occurs and to 1 if no borrow occurs.

The overflow (OV) bit is set by all instructions except Extend Sign, Multiply Step, and Divide Step, and compares to indicate arithmetic overflow (it is set to 1 when the signed result of an operation cannot be represented by 32 bits). The Extend Sign and Multiply Step instructions do not affect this bit, and the Divide Step instruction sets it according to a divide condition.

The extended operations incorporate the state of the C0 bit into the result. The extended add instructions, AE and AEI, cause the value of the C0 bit to be added to the sum of the two operands. In the extended subtract instruction, SE, the value of the first operand is added to the ones complement of the second operand, and to this result is added the value of C0 bit.

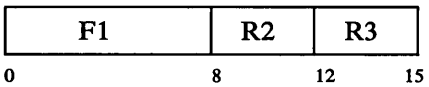
All arithmetic instructions are unprivileged.

Add (A)



The contents of registers R2 and R3 are added, and the result placed into register R2. Condition status bits LT, EQ, GT, C0 and OV are affected.

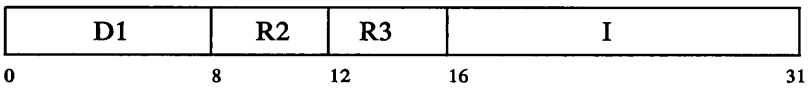
Add Extended (AE)



The content of register R2, the content of register R3, and the value of condition status bit C0 are summed, and the result placed into register R2. Condition status bits LT, EQ, GT, C 0, and OV are affected.

Note: This allows multiple precision addition.

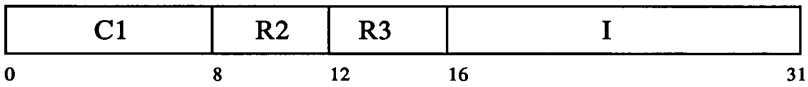
Add Extend Immediate (AEI)



The sign-extended I-field, the content of register R3, and the value of condition status bit C0 are summed and the result placed in register R2. Condition status bits LT, EQ, GT, C0, and OV are affected.

Note: This allows multiple precision addition.

Add Immediate (AI)

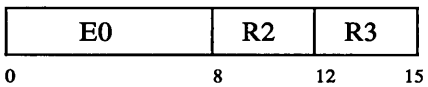


The sign extended I-field, is added to the content of register R3 and the result placed in register R2. Condition status bits LT, EQ, GT, C0, and OV are affected.

Add Immediate Short (AIS)

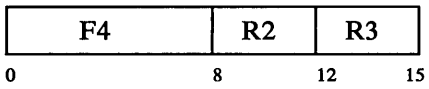
The I-field, extended on the left with 28 zeroes, is added to the content of register R2, and the result placed in register R2. Condition status bits LT, EQ, GT, C0 and OV are affected.

Absolute (ABS)



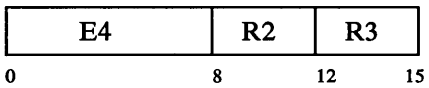
The content of register R2 is replaced by the absolute value of the content of register R3. Condition status bits LT, EQ, GT, C0 and OV are affected. Normally, only condition status bits EQ or GT are set to 1 according to the result. If register R3 contains the maximum negative number, for which there is no equivalent positive number, then the content of register R2 is set equal to the content of register R3, and the condition status bits LT and OV are set to 1.

Ones Complement (ONEC)



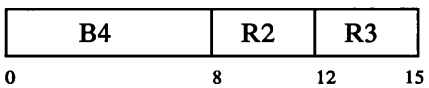
The content of register R2 is replaced by the ones complement of the content of register R3. Condition status bits LT, EQ, and GT are affected.

Twos Complement (TWOC)



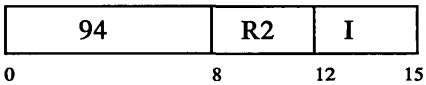
The content of register R2 is replaced by the twos complement of the content of register R3. Condition status bits LT, EQ, GT, C0, and OV are affected.

Compare (C)



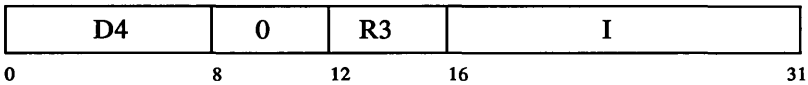
The contents of registers R2 and R3, both treated as 32-bit signed algebraic quantities, are compared. Condition status bits LT, EQ and GT are affected. Condition status bits LT and GT are set according to the true relative algebraic magnitudes of the contents of registers R2 and R3. LT is set if the content of register R2 is algebraically less than the content of register R3. GT is set if the content of register R2 is algebraically greater than the content of register R3. Condition status bit EQ is set if the content of register R2 equals the content of register R3.

Compare Immediate Short (CIS)



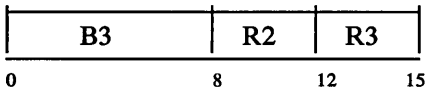
The content of register R2 is compared to the I-field extended on the left with 28 zeroes. Condition status bits LT, EQ, and GT are affected. Condition status bits LT and GT are set according to the true algebraic magnitudes of register R2 and the I- field. The LT bit is set if the content of register R2 is algebraically less than the I-field extended on the left with 28 zeroes. The GT bit is set if the content of register R2 is greater than the I-field extended on the left with 28 zeroes. The EQ bit is set if the content of register R2 equals the I-field extended on the left with 28 zeroes.

Compare Immediate (CI)



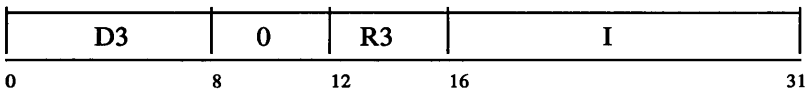
The content of register R3 is compared to sign-extended I-field. Condition status bits LT, EQ, and GT are affected. Condition status bits LT and GT are set according to the true relative algebraic magnitudes of register R3 and I-field. The LT bit is set if the content of register R3 is algebraically less than the sign-extended I-field, and the GT bit is set if the content of register R3 is greater than the sign- extended I-field. The EQ bit is set if the content of register R3 equals the sign-extended I-field.

Compare Logical (CL)



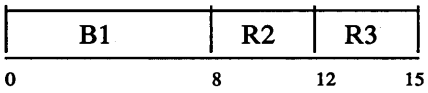
The contents of registers R2 and R3, both treated as 32-bit unsigned quantities, are compared. Condition status bits LT, EQ and GT are affected. Condition status bits LT and GT are set according to the relative unsigned magnitudes of the contents of registers R2 and R3. LT is set if the content of register R2 is logically less than the content of register R3, and GT is set if the content of register R2 is logically greater than the content of register R3. Condition status bit EQ is set if the content of register R2 equals the content of register R3.

Compare Logical Immediate (CLI)



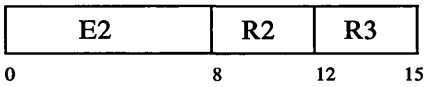
The content of register R3 is compared to the sign extended I-field. Condition status bits LT, EQ and GT are affected. Condition status bits LT and GT are set according to the relative unsigned magnitudes of register R3 and sign extended I-field. The LT bit is set if the content of register R3 is logically less than the sign extended I-field. The GT bit is set if the register R3 is greater than the sign extended I-field. The EQ bit is set if the content of register R3 equals the sign extended I-field.

Extend Sign (EXTS)



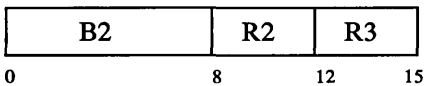
The content of the lower half of register R2 is replaced by the lower half of register R3. Bits 0-15 of register R2 are set equal to bit 16. Condition status bits LT, EQ and GT are affected.

Subtract (S)



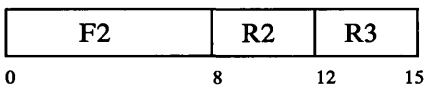
The content of register R3 is subtracted from the content of register R2, and the result placed into register R2. Condition status bits LT, EQ, GT, C0 and OV are affected.

Subtract From (SF)



The content of register R2 is subtracted from the content of register R3, and the result placed in register R2. Condition status bits LT, EQ, GT, C0 and OV are affected.

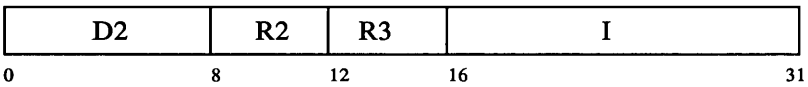
Subtract Extended (SE)



The ones complement of the content of register R3 is added to the content of register R2, and the result is added to the value of Condition status bit C0. The result is placed in register R2. Condition status bits LT, EQ, GT, C0 and OV are affected.

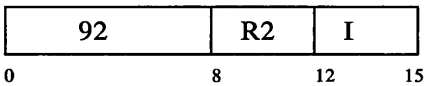
Note: This allows multiple precision subtraction.

Subtract From Immediate (SFI)



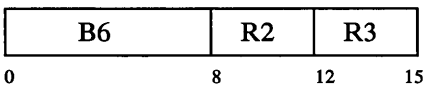
The content of register R2 is replaced by the content of register R3 subtracted from the sign extended I-field. The Condition status bits LT, EQ, GT, C0 and OV are affected.

Subtract Immediate Short (SIS)



The content of register R2 is replaced by the I-field subtracted from the content of register R2. For the subtraction, the I-field is extended on the left with 28 zeroes. Condition status bits LT, EQ, GT, C0, and OV are affected.

Divide Step (D)



The content of register R3 is added to or subtracted from (R2)// (bit of MQ) depending on whether the signs of registers R2 and R3 disagree or agree. The 32 rightmost bits of the sum replace the content of register R2. The MQ is shifted left one position, and bit 31 of the MQ is set to 1 if and only if the sign of the 33-bit result equals the sign of register R3. Condition status bit C0 is set to 1 if the sign of the 33 bit result equals the sign of the content of register R3. Bit OV is set to 1 if the sign of the 33-bit result equals the sign of the content of register R2.

Note: The Divide Step instruction can be used to construct algorithms for dividing one number by another. The following example describes an algorithm for dividing a 32-bit dividend by a 32-bit divisor. These operands are in twos complement representation.

Example: Divide X by Y giving quotient Q and remainder R where X, Y, Q and R are 32-bit numbers and Y is not equal to zero, plus one, or minus one.

Initial Conditions: General-purpose register R2 should be set to the propagated sign of X (zero if X is non-negative, minus one if X is negative). This can be accomplished by loading R2 with X and executing a SARI16 R2, 15 instruction. Load Y into general-purpose register R3. IX should then be loaded into MQ.

Algorithm: The Divide Step instruction should be issued with operands R2 and R3 32 times. If at this point the signs of R2 and R3 differ, add the content of R3 to the content of R2 replacing the content of R2. After this test and possible modification of R2, R2 contains the preliminary remainder. The MQ contains the 32 rightmost bits of the preliminary quotient. The final quotient and remainder are either equal to the preliminary quotient and remainder, or are found by adding one to the preliminary quotient and subtracting the divisor, R3, from the preliminary remainder.

Proof: The Divide Step instruction supports a nonrestoring division algorithm. Division is accomplished by repetitive subtraction. The first time, only the sign of the dividend extended to an appropriate width participates in the subtraction. On each subsequent repetition, an additional dividend bit is included to the right of the result of the previous repetition; this has the effect of halving the divisor.

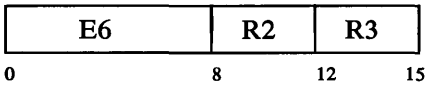
Because the division involves binary numbers, the divisor can be subtracted from the current minuend either 1 or 0 times. If it is 1, the appropriate quotient bit is 1. If it is 0, the quotient bit is 0; however, the subtraction has already been performed. Instead of adding the divisor back at this point, half the divisor is added at the next repetition, since the result of subtracting the divisor and adding half the divisor is the same as subtracting half the divisor.

When all dividend bits have been used, and the signs of the divisor and remainder differ, restoration must be done for the last step, and so the divisor is added back into the remainder without changing the quotient. This produces the preliminary quotient and remainder.

The actual algorithm depends on the signs of the divisor and the dividend at each step. These signs also determine whether the initial step is addition or subtraction.

The above algorithm can be modified for dividing a 64-bit dividend contained in R2//MQ. If the initial value of (R2)//(bit 0 of MQ) exceeds the divisor in magnitude, then divide overflow will occur. This condition can be determined by testing for OV=1 after execution of the first Divide Step instruction.

Multiply Step (M)



The incomplete product of the content of register R3 and bits 30 and 31 of the MQ register is formed in (R2)//MQ. A 34-bit sum is formed in accordance with the table below. The MQ is algebraically shifted right two positions, with the two rightmost bits of the sum replacing bits 0 and 1 of the MQ. The content of register R2 is replaced by the 32 leftmost bits of the sum. Condition status bit C0 is set to the complement of bit 30 of the MQ before the shift.

Condition Status Bit C0	**MQ Bit 30**	**MQ Bit 31**	**Algebraic Sum**
0	0	0	(R2) + (R3)
0	0	1	(R2) + 2*(R3)
0	1	0	(R2) - (R3)
0	1	1	(R2) + 0
1	0	0	(R2) + 0
1	0	1	(R2) + (R3)
1	1	0	(R2) + 2*(R3)
1	1	1	(R2) - (R3)

Multiply Step Operations

The Multiply Step instructions can be used to construct algorithms for multiplying two numbers together. The following example describes an algorithm for multiplying a 32-bit multiplicand by a 16-bit multiplier. The operands are in twos complement representation.

Example: Multiply X by Y giving Z where X is a 32-bit number and Y is a 16-bit number.

Initial Conditions: Load X into general-purpose register R3; load Y into the MQ; set the content of general-purpose register R2 to 0; set condition status bit C0 to 1. R2 and C0 can be initialized simultaneously by executing an S R2, R2 instruction.

Algorithm: Issue the Multiply Step instruction with operands R2 and R3 eight times.

Result: The 16 rightmost bits of the product Z are in the MQ; the 32 leftmost bits are in register R2.

Proof: The 16-bit multiplier Y can be expressed as the sum of 16 terms of the form:

$$y_i 2^i \text{ where } y \text{ equals 0 or 1 and } i = 0,1,2...15$$

or eight terms of the form:

$$(y_{2i} + 2*y_{2i+1}) * 4^i$$

where y_{2i} and y_{2i+1} equal 0 or 1 and i = 0,1...,7.

The Multiply Step instruction accumulates a partial sum in register R2 and the leftmost bits of the MQ; a condition status bit C0 equal to 0 indicates a carry. The instruction provides four cases for the parenthesized factor when there is no carry into the term and four cases when there is a carry.

$y_{2i} + 2*y_{2i+1}$	**Carry**	**Value**
0	No	(R2)
1	No	(R2) + (R3)
2	No	(R2) + 4*(R3) - 2*(R3)
3	No	(R2) + 4*(R3) - (R3)
0	Yes	(R2) + (R3)
1	Yes	(R2) + 2*(R3)
2	Yes	(R2) + 4*(R3) - (R3)
3	Yes	(R2) + 4*(R3)

In the parenthesized factor, y_{2i} is the value of MQ bit 31, and y_{2i+1} is the value of MQ bit 30. Whenever MQ bit 30 is 1, the term 4*(R3) appears. This is a carry into the next partial sum.

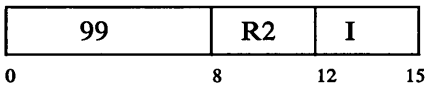
The Multiply Step instruction places the rightmost two bits of the partial sum in vacated MQ bits (bits 0 and 1). This provides the 4^i factor since it has the effect of multiplying the remaining bits of the multiplier by four.

Logical Operations

The logical operations treat the contents of the general-purpose register as 32-bit unsigned integers except for the instruction Count Leading Zeroes (CLZ), which is applied to the lower half of a register. All logical operations except CLZ set condition status bits LT, EQ and GT according to the algebraic value expressed in twos complement representation. If the result is a negative value, LT is set to 1; if it is 0, EQ is set to 1; if it is positive and not 0, GT is set to 1. The condition status is unaffected by the Count Leading Zeroes instruction.

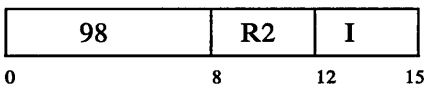
All logical instructions are unprivileged.

Clear Bit Lower Half (CLRBL)



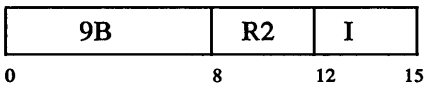
A bit in the lower half of register R2 is set to 0, where the bit is specified by the immediate I-field. Condition status bits LT, EQ, and GT are affected.

Clear Bit Upper Half (CLRBU)



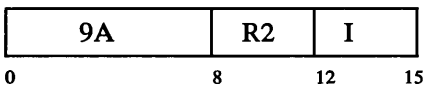
A bit in the upper half of register R2 is set to 0, where the bit is specified by the immediate I-field. Condition status bits LT, EQ, and GT are affected.

Set Bit Lower Half (SETBL)



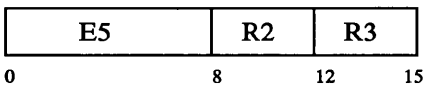
A bit in the lower half of register R2 is set to 1, where the bit is specified by the immediate I-field. Condition status bits LT, EQ and GT are affected.

Set Bit Upper Half (SETBU)



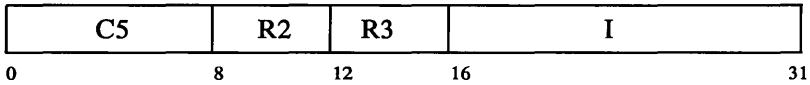
A bit in the upper half of register R2 is set to 1, where the bit is specified by the immediate I-field. Condition status bits LT, EQ and G T are affected.

AND (N)



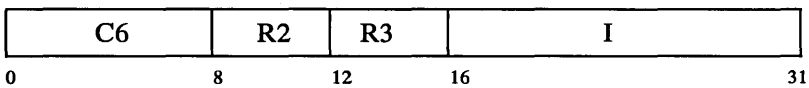
The AND of the contents of registers R2 and R3 replaces the content of the register specified by R2. Condition status bits LT, EQ and GT are affected.

AND Immediate Lower Half Extended Zeroes (NILZ)



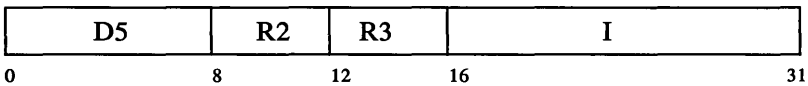
The AND of the I-field extended on the left with 16 zeroes and the content of register R3 replace the content of register R2. Condition status bits LT, EQ and GT are affected.

AND Immediate Lower Half Extended Ones (NILO)



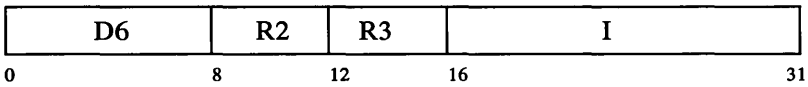
The AND of the I-field extended on the left with 16 ones and the content of register R3 replace the content of register R2. Condition status bits LT, EQ and GT are affected.

AND Immediate Upper Half Extended Zeroes (NIUZ)



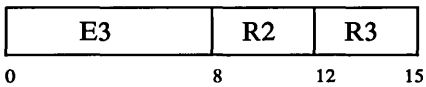
The AND of the I-field extended on the right with 16 zeroes and the content of register R3 replace the content of register. Condition status bits LT, EQ and GT are affected.

AND Immediate Upper Half Extended Ones (NIUO)



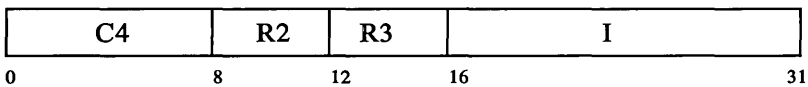
The AND of the I-field extended on the right with 16 ones and the content of register R3 replace the content of register R2. Condition status bits LT, EQ and GT are affected.

OR (O)



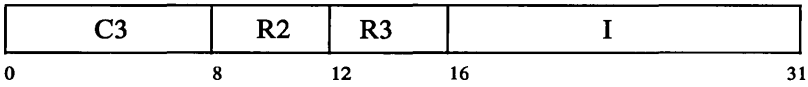
The OR of the contents of registers R2 and R3 replaces the content of register R2. Condition status bits LT, EQ and GT are affected.

OR Immediate Lower (OIL)



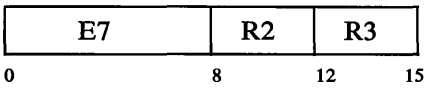
The OR of the I-field extended on the left with 16 zeroes and the content of register R3 replaces the content of register R2. Condition status bits LT, EQ and GT are affected.

OR Immediate Upper (OIU)



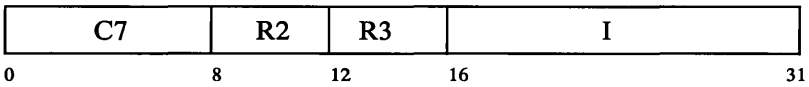
The OR of the I-field extended on the right with 16 zeroes and the content of register R3 replaces the content of register R2. Condition status bits LT, EQ and GT are affected.

Exclusive OR (X)



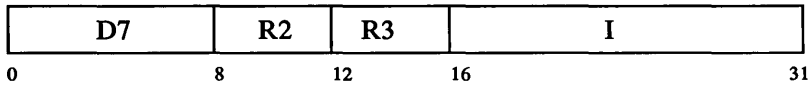
The EXCLUSIVE OR of the contents of registers R2 and R3 replaces the content of register R2. Condition status bits LT, EQ and GT are affected.

Exclusive OR Immediate Lower Half (XIL)



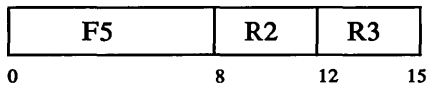
The EXCLUSIVE OR of the I-field extended on the left with 16 zeroes and the content of register R3 replaces the content of register R2. Condition status bits LT, EQ and GT are affected.

Exclusive OR Immediate Upper Half (XIU)



The EXCLUSIVE OR of the I-field extended on the right with 16 zeroes and the content of register R3 replaces the content of register R2. Condition status bits LT, EQ and GT are affected.

Count Leading Zeroes (CLZ)



The content of register R2 is replaced by the binary representation of the number of leading zeroes in the lower half of register R3 (that is, the number of zeroes to the left of the leftmost 1 bit in the lower half of register R3).

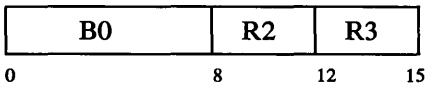
Note: If the lower half of register R3 is equal to 0, the content of register R2 is replaced by the binary representation of 16.

Shifts

Shift instructions operate on either the content of a register or a register half. Immediate form shifts specify a shift amount of 0 to 31 bits to the left or right based on the value of the immediate field. Indirect shifts specify a shift amount of 0 to 63 bits to the left or right based on the low-order six bits of register R3. A shift amount greater than 31 bits results in a 32-bit shift. All shifts set the condition status bits LT, EQ and GT according to the resultant algebraic value returned to the register. All instructions except the shift algebraic right instructions supply zeroes to the vacated bit positions.

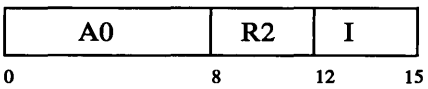
All shift instructions are unprivileged.

Shift Algebraic Right (SAR)



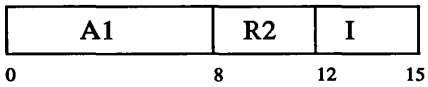
The content of register R2 is shifted right the number of bit positions specified by bits 26-31 of register R3. Bits equal to the original sign bit (bit 0) are supplied to the vacated high-order positions. Condition status bits LT, EQ and GT are affected.

Shift Algebraic Right Immediate (SARI)



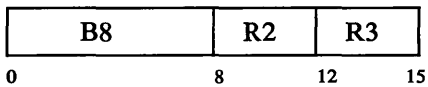
The content of register R2 is shifted right the number of bit positions specified by the I-field. Bits equal to the original sign bit (bit 0) are supplied to the vacated high-order positions. Condition status bits LT, EQ and GT are affected.

Shift Algebraic Right Immediate Plus Sixteen (SARI16)



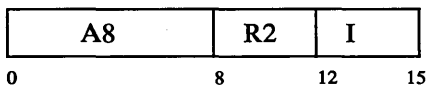
The content of the register R2 is shifted right the number of bit positions specified by the I-field plus 16. Bits equal to the original sign bit (bit 0) are supplied to the vacated high-order positions. Condition status bits LT, EQ and GT are affected.

Shift Right (SR)



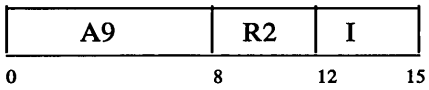
The content of register R2 is shifted right the number of bit positions specified by bits 26-31 of register R3. Zeroes are supplied to the vacated high-order positions. Condition status bits LT, EQ and GT are affected.

Shift Right Immediate (SRI)



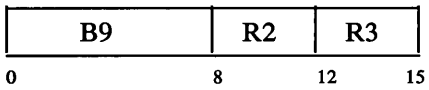
The content of register R2 is shifted right the number of bit positions specified by the I-field. Zeroes are supplied to the vacated high-order positions. Condition Status bits LT, EQ and GT are affected.

Shift Right Immediate Plus Sixteen (SRI16)



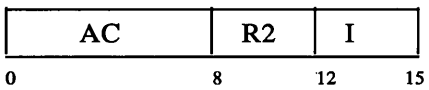
The content of register R2 is shifted right the number of bit positions specified by the I-field plus 16. Zeroes are supplied to the vacated high-order positions. Condition status bits LT, EQ and GT are affected.

Shift Right Paired (SRP)



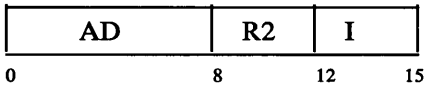
The content of the register R2, shifted right the number of bit positions specified by bits 26-31 of register R3 with zeroes supplied to the vacated high-order positions, is placed in the twin register R2. The content of register R2 is not affected. Condition status bits LT, EQ and GT are affected.

Shift Right Paired Immediate (SRPI)



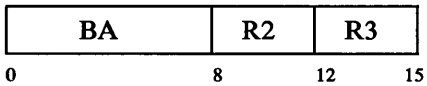
The content of register R2, shifted right the number of bit positions specified by the I-field with zeroes supplied to the vacated high-order positions, is placed in the twin register R2. The content of register R2 is not affected. Condition status bits LT, EQ and GT are affected.

Shift Right Paired Immediate Plus Sixteen (SRPI16)



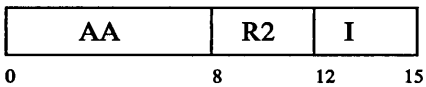
The content of register R2, shifted right the number of bit positions specified by the I-field plus 16 with zeroes supplied to the vacated high-order positions is placed, in the twin of register R2. The content of register R2 is not affected. Condition status bits LT, EQ and GT are affected.

Shift Left (SL)



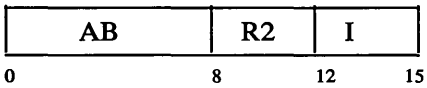
The content of register R2 is shifted left the number of bit positions specified by bits 26-31 of register R3. Zeroes are supplied to the vacated low-order positions. Condition status bits LT, EQ and GT are affected.

Shift Left Immediate (SLI)



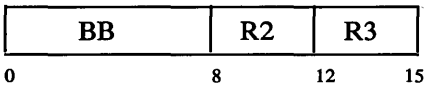
The content of the register R2 is shifted left the number of bit positions specified by the I-field. Zeroes are supplied to the vacated low-order positions. Condition status bits LT, EQ and GT are affected.

Shift Left Immediate Plus Sixteen (SLI16)



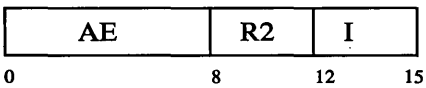
The content of register R2 is shifted left the number of bit positions specified by the I-field plus 16. Zeroes are supplied to the vacated low-order positions. Condition Status bits LT, EQ and GT are affected.

Shift Left Paired (SLP)



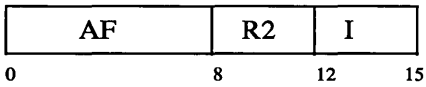
The content of register R2 shifted is left the number of bit positions specified by bits 26-31 of register R3 with zeroes supplied to the vacated low-order positions is placed in the twin of register R2. The content of register R2 is not affected. Condition status bits LT, EQ and GT are affected.

Shift Left Paired Immediate (SLPI)



The content of register R2 is shifted left the number of bit positions specified by the I-field with zeroes supplied to the vacated low-order positions is placed in the twin of register R2. The content of register R2 is not affected. Condition status bits LT, EQ and GT are affected.

Shift Left Paired Immediate Plus Sixteen (SLPI16)



The content of the register R2 is shifted left the number of bit positions specified by the I-field plus 16 with zeroes supplied to the vacated low order positions is placed in the twin of register R2. The content of register R2 is not affected. Condition status bits LT, EQ and GT are affected.

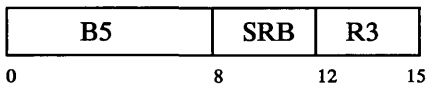
System Control

The system control instructions provide a means of examining and manipulating the state of certain processor facilities. This is done through two subclasses of instructions. The first subclass operates on the contents of the system control registers. These instructions allow the reading or writing of any SCR or the setting or clearing of any of the low-order 16 bits of the SCR. A second subclass of instructions within this class provides the necessary software interface to the interrupt facility described in "Interrupts" on page 11-19.

The instructions that deal with the SCRs provide a general capability of operating on any of the SCRs. However, because of the definition of certain SCRs, not every operation on an SCR gives a predictable result. Moreover, bits in any SCR that have been specified as reserved bits cannot be used in a predictable manner. These exceptional cases are specified along with the instruction definitions. Finally, all SCRs except the ICS (SCR 14) are dynamically changed by the processor, often asynchronously to instruction sequencing. Therefore, a read of an SCR following a write does not necessarily get the same data that was written.

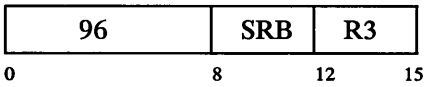
Only certain system control instructions are unprivileged. The unprivileged instructions are MTS, MFS, SETSB, and CLRSB when the SCR referred to by these instructions is the MQ or CS, and the SVC instruction. An attempt to execute any other system control instruction in unprivileged state causes the privileged instruction exception bit in the program check status to be set and a program check to occur. Refer to "Program-Check Status" on page 11-87 for a description of the program check status.

Move to SCR (MTS)



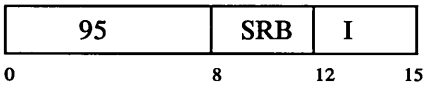
The content of system control register SRB is replaced by the content of register R3. Any reserved bits in the specified SCR are not set to predictable values. If the specified SCR is the IAR (SCR 13), the results of this instruction are unpredictable.

Move from SCR (MFS)



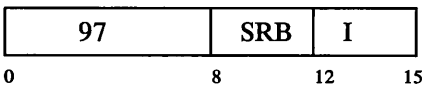
The content of register R3 is replaced by the content of system control register SRB. The bits of register R3 corresponding to reserved bits of the specified SCR are set to unpredictable values. If the specified SCR is the IAR (SCR 13), the value that is loaded into register R3 is the address of the instruction immediately following the MFS instruction in system memory.

Clear SCR Bit (CLRSB)



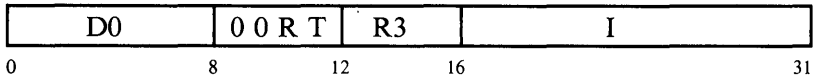
A bit in the lower half of system control register SRB is set to 0, where the bit is selected by the immediate I-field. If the selected bit of the SCR is a reserved bit, it is not set to a predictable value. If the specified SCR is the IAR (SCR 13), the results of this instruction are unpredictable.

Set SCR Bit (SETSB)



A bit in the lower half of system control register SRB is set to 1, where the bit is selected by the immediate I-field. If the selected bit of the SCR is a reserved bit, it is not set to a predictable value. If the specified SCR is the IAR (SCR 13), the results of this instruction are unpredictable.

Load Program Status (LPS)



The content of the IAR (SCR 13) is replaced by the word in system memory addressed by 0/(R3) plus the sign extended I-field. The content of the ICS (SCR 14) is replaced by the content of the system memory halfword addressed by 0/(R3) plus the sign extended I-field plus four. The content of the CS (SCR 15) is replaced by the content of the system memory halfword addressed by 0/(R3) plus the sign extended I-field plus six. Any reserved bits in the SCRs are set to unpredictable values. If the processor is on the machine check level (see "Machine-Check Status" on page 11-86) when the LPS is executed, the content of the MCS is set to 0. If the processor is on the program check level (see "Program-Check Status" on page 11-87) when the LPS is executed, the content of the PCS is set to 0.

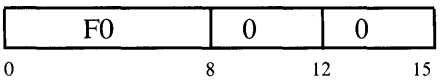
If bit 10 of this instruction is a 1, pending memory operations are restarted before instruction execution is resumed. The exception control register (SCR 9) contains the count and system memory address of the exception information for operations to be restarted.

If bit 11 of this instruction is a 1, interrupts remain pending until the target instruction of the LPS instruction has been executed. If bit 11 is 0, interrupts may occur after the LPS instruction is executed.

Programming Note:

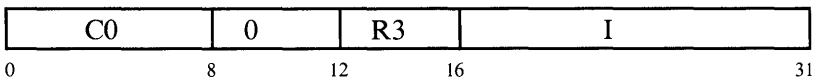
The LPS instructions may be used to return from an interrupt. The LPS instruction may also be used to trace instruction execution. This is accomplished by setting a bit in the IRB to generate an interrupt request before executing the LPS instruction. The bit that is set should have a corresponding interrupt request priority greater than the processor priority that is loaded by the LPS instruction. If the interrupt mask that is loaded by the LPS is 0, and if bit 11 of the LPS instruction is a 1, an interrupt occurs after the target instruction of the LPS has been executed.

Wait (WAIT)



The processor enters the wait state. When the processor is in the wait state, it does not execute any instructions nor make any memory accesses. The processor exits the wait state on a interrupt, error, or power-on reset.

Supervisor Call (SVC)



The content of the IAR (SCR 13) is stored into the word in system memory beginning at address X'190'. The content of the ICS (SCR 14) is stored into the halfword in system memory beginning at address X'194'. The content of the CS (SCR 15) is stored into the halfword in system memory beginning at address X'196'. The low-order 16-bits of the 32-bit sum 0/(R3) + 0[16]//I is stored into the halfword in system memory beginning at address X'19E'. The content of the IAR (SCR 13) is replaced by the word in system memory beginning at address X'198'. The content of the ICS (SCR 14) is replaced by the content of the halfword in system memory beginning at address X'1 9C'. Any reserved bits in the IAR and the ICS are set to unpredictable values.

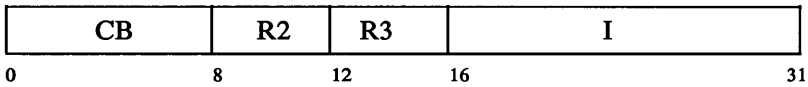
Input/Output

Programmed I/O (PIO) instructions are used to transfer data between the general-purpose registers and registers in the memory management unit.

All I/O addresses are considered to be device addresses. The upper byte of the I/O address is checked to be 0. A nonzero high-order byte in the I/O address causes a program check.

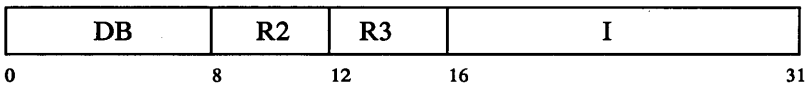
All PIO instructions are unprivileged. Each I/O device determines whether it is a privileged or unprivileged device. Privileged I/O devices accept I/O commands from the processor only when the processor is in privileged state. An attempt to access a privileged I/O device from unprivileged state causes the data address exception bit in the program check status to be set and a program check to occur. See "Privileged and Unprivileged States" on page 11-12 for a description of privileged I/O device connection, and "Program-Check Status" on page 11-87 for a description of the program check status.

Input/Output Read (IOR)



The content of register R2 is replaced by data transferred from the I/O device selected by the effective address 0/(R3) + 0[16]//I. Bits 8-31 of the 32-bit effective address are interpreted as the I/O device address. Bits 0-7 of the effective address must be 0.

Input/Output Write (IOW)



The content of register R2 is transferred to the I/O device selected by the effective address 0/(R3) + 0[16]//I. Bits 8-31 of the 32-bit effective address are interpreted as the I/O device address. Bits 0-7 of the effective address must be 0.

I/O Capability

The system processor provides two capabilities for controlling I/O operations: programmed I/O (PIO), and I/O interrupts.

Programmed I/O

Two programmed I/O (PIO) instructions (IOR and IOW) provide I/O operations that are synchronous to the program. For each PIO instruction executed, a 24-bit I/O address field is sent to an I/O device, and data is transferred between the I/O device and a general-purpose register. These I/O instructions are used to access memory management unit registers.

I/O Interrupt Requests

I/O interrupt requests report asynchronous events. Each of these interrupt requests is assigned one of seven priority levels. Processor logic allows I/O interrupts (unless masked) on a priority basis. The interrupt facility is described in "Interrupts" on page 11-19.

RAS Facilities

RAS Facilities provide for:

- Detection of processor errors
- Detection and isolation of program-related errors
- Decreased exposure to data loss and error situations.

Internal Diagnostics

The processor executes an internal microcode routine to perform register initialization when a processor reset occurs. Successful completion of the register initialization routine normally indicates that the processor is functional for instruction execution. The internal microcode diagnostic does not verify any internal bus functions, or other system components. No other system components are verified by the internal diagnostic.

Machine-Check Errors

Machine-check errors are those errors that are most often caused by hardware malfunctions.

Machine-Check Error Handling

Upon the detection of a machine-check error condition, other than an I/O trap, all current processor activity is halted, regardless of that activity. If the detected error is an I/O trap, the processor completes its current activity before servicing the error. I/O traps are reported by activating the '-TRAP' input. The processor then takes one of two courses of action, depending on the value contained in the check stop mask.

If the check stop mask has a value of 0, the processor enters the check stop state when any machine check error is detected (including an error reported by -TRAP). This preserves the state of internal processor latches for inspection by a support processor. The I/O pin '-FAIL' goes active to indicate a failure.

If the check stop mask has a value of 1, and a machine-check error is detected other than one caused by the '-TRAP' interrupt input, a reset packet is sent on the processor channel to clear any pending processor channel operations. If the machine-check interrupt is caused by the '-TRAP' interrupt input, the reset packet is not sent on the processor channel. If the check stop mask has a value of 1, the processor saves the current program status in the old program status location in the Machine Check Old and New PS Pairs (beginning at location X'170'). The program status for servicing the error is then loaded from the new program status location except for the condition status, and the processor attempts to continue execution.

The machine-check routine must execute a Load Program Status (LPS) instruction to return from the machine check error.

Machine-Check Status

The machine-check status (MCS) provides a means for reporting hardware malfunctions. Information is provided to assist an error-servicing routine in determining the type and source of the error.

The MCS is an 8-bit field in system control register 11. Upon the detection of a machine-check error, appropriate bits of the MCS are set to ones (except for the parity check bit (bit 18) which is set whenever the processor receives a reply with invalid data parity).

The MCS is defined as follows:

Bit 16 Processor Channel Check. Set to 1 when a device on the processor channel detects invalid parity on a processor-generated transfer over an abnormally large number of retries. This bit is also set when the processor generates an interrupt to report an processor channel retry that successfully corrected a parity error (See "Interrupt Control Status" on page 11-24).

Bit 17 Reserved.

Bit 18 Parity Check. Set to 1 whenever the processor receives a reply on the processor channel with invalid data parity. This bit is set whether or not a machine check occurs.

Bit 19 Instruction Timeout. Set to 1 when the processor fails to receive an expected reply to an instruction fetch.

Bit 20 Data Timeout. Set to 1 when the processor fails to receive an expected reply to a data fetch.

Bit 21 Processor Channel Timeout. Set to 1 whenever the processor has been unable to transfer a request on the processor channel over an abnormally large number of cycles, and no parity errors have been signalled. The request may be unsuccessful due to busy responses or unsuccessful arbitration.

Bit 22 I/O Trap. Set to 1 when an I/O device signals a trap condition.

Bit 23 Reserved.

The MCS is cleared when a Load Program Status (LPS) instruction is executed to return from the machine check level.

The MCS provides a summary of processor conditions that are present when a machine-check error is detected. Thus, multiple bits of the MCS can be set on detection of an error. For example, if bits 16, 19, and 20 of the MCS are set, the processor failed to receive a reply to both an instruction and a data fetch. In addition, a device on the processor channel has detected invalid parity on a processor-generated request. In this case, invalid parity on the request has prevented the processor from successfully transferring both instruction and data requests.

Program-Check Errors

Program-check errors are those errors that are usually caused by software errors.

Program-Check Error Handling

On the detection of a program-check error condition, the processor completes its current activity (such as, instruction, timer) unless that activity caused the program-check condition. The processor then saves the current program status in the old program status location in the Program Check Old and New PS Pair (beginning at location X'180'). The program status for servicing the error is loaded from the new program status location, except for the condition status.

The program-check routine must execute a Load Program Status (LPS) instruction to return from the program-check error.

Program-Check Status

The Program-Check Status (PCS) provides a means for reporting certain programming errors. Reported program-check errors include:

- Attempted execution of an unassigned or unimplemented operation code
- Attempted execution of a privileged instruction with the unprivileged state bit (bit 21 of SCR 14) being set at 1
- An improper data condition which is detected by the execution of a trap instruction
- Attempted access of an invalid memory location.

The PCS is an 8-bit field in system control register 11. On the the detection of a program-check error, all bits of the PCS are set to zeros. The appropriate bits of the PCS are then set to ones.

The PCS is defined as follows:

Bit 24 Program check with known origin. Set to 1 when a program check occurs and the location of the failing instruction is determined from the IAR in the old program status.

Bit 25 Program check with unknown origin. Set to 1 when a program check occurs and the location of the failing instruction cannot be determined from the IAR in the old program status.

Bit 26 Program Trap. Set to 1 when a trap exception condition is generated by a trap instruction.

Bit 27 Privileged Instruction Exception. Set to 1 when the processor attempts to execute a privileged instruction and the unprivileged state bit (bit 21 of SCR 14) is set to 1.

Bit 28 Illegal Operation Code. Set to 1 when the attempted execution of an unassigned or unimplemented operation code or illegal subject of a branch with execute instruction is detected.

Bit 29 Instruction Address Exception. Set to 1 when no device on the processor channel responds to a processor instruction fetch request, or when a reply to an instruction fetch is accompanied by an invalid address indication, or when the processor is in real mode and the high order address byte is not zero.

Bit 30 Data Address Exception. Set to 1 when no device on the processor channel responds to a processor data request, or when a device on the processor channel responds to a data request with an invalid address indication. This bit is also set when an access to a privileged I/O device is attempted from unprivileged state. It is also set to 1 if the high order address byte is not 0 and the processor is in real mode or the request is an I/O Read or I/O Write.

Bit 31 Reserved.

The PCS is cleared by the Load Program Status (LPS) instruction.

The detection of a program trap condition sets bits 24 and 26 to ones. The IAR in the old program status contains the address of the trap instruction.

The detection of an illegal branch with execute subject instruction sets both bits 24 and 28 to ones. The IAR in the old program status contains the address of the branch with execute instruction.

If a data address exception occurs, bit 30 of the PCS is set to 1, and either bit 24 or 25 is set to 1 to indicate the meaning of the IAR in the old program status. If bit 24 is set to 1, the IAR in the old program status contains the address of the instruction that attempted to access the invalid memory location. If the subject instruction of a branch with execute attempts to access an invalid memory location, the IAR in the old program status contains the address of the branch with execute instruction. If bit 25 of the PCS is set to 1, the IAR in the old program status contains the address of the instruction that was executing when the data address exception was detected. If this instruction required the data which was accessed at the invalid address, it did not complete successfully.

Figure 11-9 provides a summary of program-check errors when address translation and memory protect are disabled.

Figure 11-10 provides a summary of program-check errors when address translation or memory protect is enabled.

	PCS Bits							
Program Check Error	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
Invalid instruction address	1	0	0	0	0	1	0	0
Invalid data address	0	1	0	0	0	0	1	0
Successful trap instruction	1	0	1	0	0	0	0	0
Privileged instruction exception	1	0	0	1	0	0	0	0
Illegal op-code	1	0	0	0	1	0	0	0

Figure 11-9. Program Check Errors (Memory Protect and Address Translation Disabled)

	PCS Bits							
Program Check Error	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
Invalid instruction address	1	0	0	0	0	1	0	0
Invalid data address	1	0	0	0	0	0	1	0
Successful trap instruction	1	0	1	0	0	0	0	0
Privileged instruction exception	1	0	0	1	0	0	0	0
Illegal op-code	1	0	0	0	1	0	0	0

Figure 11-10. Program Check Errors (Memory Protect or Address Translation Enabled)

Simultaneous Program-Check and Machine-Check Errors

Certain hardware error conditions can result in both a program-check and a machine-check error, as the result of a single request. For example, a memory controller may generate both an exception reply and activate the '-TRAP' input to report an uncorrectable memory error (uncorrectable ECC error or parity error). The exception reply causes a program check, and the -TRAP causes a machine check. The exception reply is necessary to prevent the processor from using bad data from the request, and requires '-TRAP' to report hardware errors at the machine-check interrupt level. If a machine-check and program-check error occur simultaneously, the processor performs both a program-check PSW swap, and then a machine- check PSW swap. To guarantee that both the program-check and machine-check interrupts are properly handled, system devices using both an exception reply and '-TRAP' must report both errors simultaneously. Devices should activate '-TRAP' when the error is detected, and send an exception indication with the reply.

Once the program-check and machine-check PSW swaps are completed by the processor, the machine-check interrupt handler is executed. The machine-check and program-check interrupt handlers must be properly designed to handle this multiple error condition. The machine-check interrupt handler can determine the reason for the interrupt by examining status bits in the MCS and status register in each system component. Once the source of the error has been isolated and logged, the machine-check interrupt handler can complete and should execute an LPS instruction to return from the machine-check interrupt. Executing an LPS on the machine-check level clears all bits in the MCS and returns to the next highest priority level. In this case, a program-check interrupt is pending so the machine-check interrupt handler returns to the program-check interrupt handler.

System software must be constructed so that sufficient information is available to the program-check interrupt handler. This determines that the program-check interrupt was due to the machine check error that was previously handled. If the machine-check interrupt handler performed all of the steps necessary to service the error, then the program-check interrupt handler executes an LPS instruction to return from the program-check level. Executing the LPS on the program-check level clears all bits in the PCS, and returns to the routine that was executing when the error was detected.

A single error causes both a machine-check and a program-check interrupt, and that both interrupts must be processed. The machine-check interrupt handler is always executed first, and returns to the program-check interrupt handler by executing an LPS instruction. Once the program-check interrupt handler completes, it returns to the routine in which the error was detected by executing an LPS. Executing an LPS from the machine-check level terminates processing on the machine-check level, clears all bits in the MCS, and returns to the next highest priority interrupt.

Multiple Occurrence of Errors

The processor will enter the check-stop state, regardless of the value in the check-stop mask, under the following conditions:

- The processor is servicing a machine-check error, and another machine-check error is detected.
- The processor is servicing a machine-check error, and a program-check error is detected.
- The processor is servicing a program-check error, and another program-check error is detected.

If the processor is servicing a program-check error and a machine-check error is detected, the machine-check error is handled as outlined in "Machine-Check Errors" on page 11-85.

Memory Management Unit

The memory management unit attaches to the processor channel and contains the logic required to interface with up to 16M-bytes of memory. The memory management unit is functionally divided into the three sections shown in Figure 11-11 on page 11-93. The function of each of these sections is described below:

1. Processor Channel Interface. This logic consists of the common front end (CFE) macro that provides the proper protocol from the processor channel to the address translation logic and memory control logic. All communication to and from the processor channel is handled by this logic.
2. Address Translation Logic. This logic provides the translation from a virtual address received from the processor channel to a real address used to access memory. This logic contains a translation look-aside buffer (TLB) organized as 2-way set associate with 16 congruence classes. This logic automatically reloads TLB entries from page tables in system memory as required. Operation and use of the address translation logic is described in "Address Translation Process" on page 11-93 through "I/O Address Assignments" on page 11-136.
3. Memory Control. This logic provides the address, data, and memory control signals that link the memory management unit to external RAM and ROS. This logic also provides ECC logic and dynamic memory refresh control.

In addition to the memory control logic provided in the memory management unit, external interface logic is contained on the processor board to adapt the memory management unit memory interface to that required for ROM and RAM.

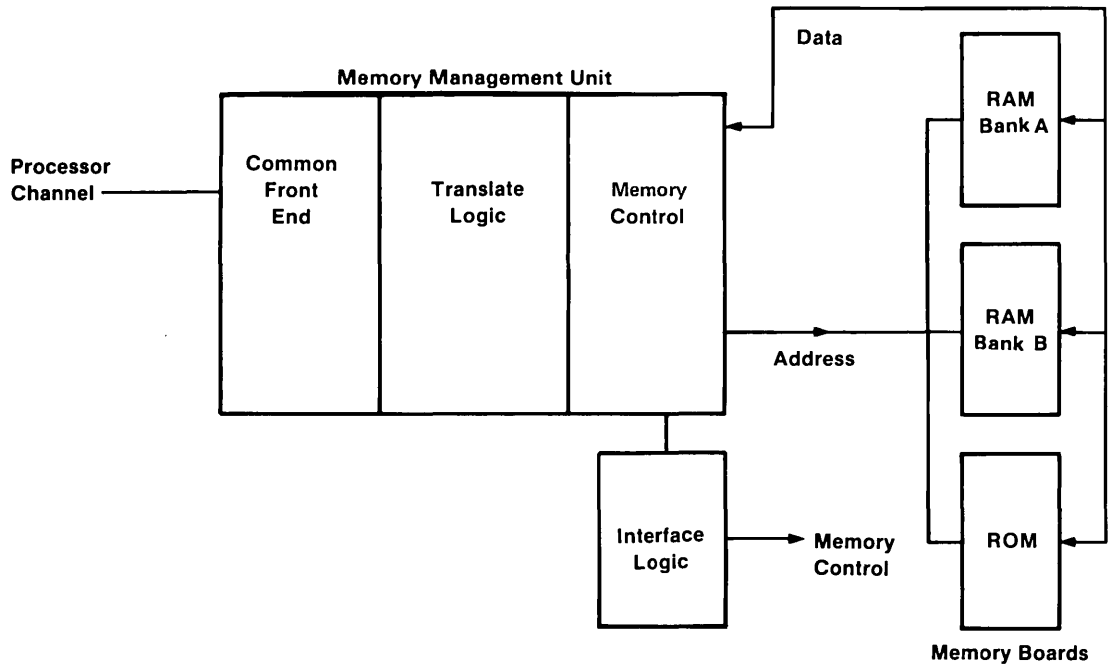


Figure 11-11. Memory Management Unit Functional Partition

Address Translation Process

Whether an address is treated as virtual (translated) or treated as real is controlled by the value of the translate mode bit on the processor channel. Each device that places a request on the processor channel controls the value of the translate mode bit for each request. If the translate mode bit is active, the address is treated as a virtual address and is translated. When the translate mode bit is inactive, the address is not translated. A virtual equal-to-real addressing mode that allows selected virtual addresses to bypass the normal address translation, memory protection, and lockbit processing is provided. Virtual equal to real mode is enabled by a bit in the translation control register (see "Translation Control Register" on page 11-117).

Address Translation Overview

The memory management unit address translation mechanism provides support for the following:

- Multiple independent virtual address spaces
- Address space size of 4 gigabytes
- Demand paging
- Page size of 2048 or 4096 bytes
- Memory protection
- Shared segments for instructions and data
- Special segment support with high resolution protection
- Real memory addressability of up to 16 megabytes
- Reference and change bits for each real page
- Hardware assist for load real address, invalidated TLB entries, and memory exception address.

Memory is treated as if it were mapped onto a single 40-bit virtual address space consisting of 4096 segments of 256 megabytes each. The 32-bit address received from the processor channel is converted to a 40-bit ("long form virtual") address by using the four high-order bits to select one of 16 segment registers, and the 12-bit contents which concatenated with the remaining 28 bits of the effective address. The translation mechanism then converts the 40-bit virtual address to a real address for memory access.

At any given instant, only 4 gigabytes of memory is addressable, namely the sixteen 256 megabyte segments specified by the 16 segment registers. This fact allows the operating system to create multiple independent virtual address spaces by loading appropriate values into the segment registers. As a limiting case, 256 completely independent 4 gigabyte address spaces can be created in this manner, although some segments (such as nucleus code) can be shared across multiple address spaces.

Memory protection similar to System 370 is provided on a 2K or 4K-byte page basis. Memory protect and fetch protect are supported, with the protect key (equivalent to the key in the S/370 PSW) specified independently for each 256 megabyte segment.

Support for special segments is provided by a set of lock bits associated with each virtual page. The lock bits effectively extend the memory protection granularity to lines of memory (128 bytes for 2K pages, or 256 bytes for 4K pages) and allow the operating system to detect and automatically journal changes to persistent variables.

Each of the 16 segments can be marked as present or not present. The memory management unit accepts a virtual address only if the selected segment register entry indicates that the segment is present. This facility allows the virtual address space to be partitioned among multiple memory management units (or other devices) on the processor channel.

Isolation of system processor and I/O device access is provided on a segment basis which allows each segment to be accessible by system processor requests only, I/O device requests only, both requests, or neither. This function can be used to protect against unauthorized access of memory by either the system processor or an I/O device (hard file).

Address Translation Hardware Operation

The incoming 32-bit effective address is first expanded to a 40-bit virtual address by concatenating a segment identifier to the effective address. The virtual address is then presented to the translation hardware for conversion to the equivalent real address. Virtual addresses are translated to a real memory address by the process described in "Segment Register Selection" on page 11-96 through "TLB Reload from System Memory Page Tables" on page 11-101.

Segment Register Selection

The high-order 4 bits of the incoming effective address index into the segment table to select one of 16 segments. A segment present bit, a system processor access protect bit, an I/O access protect bit, a 12-bit segment identifier, a special segment bit, and a key bit are obtained from the selected segment register. The segment present bit indicates the corresponding segment is assigned to the memory management unit as described in "Segment Registers" on page 11-127. The system processor access protect bit, the I/O protect bit, the special segment bit, and the key bit are used for access validation as described in "Memory Access Control" on page 11-108. The 12-bit segment identifier is used for formation of the virtual address. Figure 11-12 shows the segment table format.

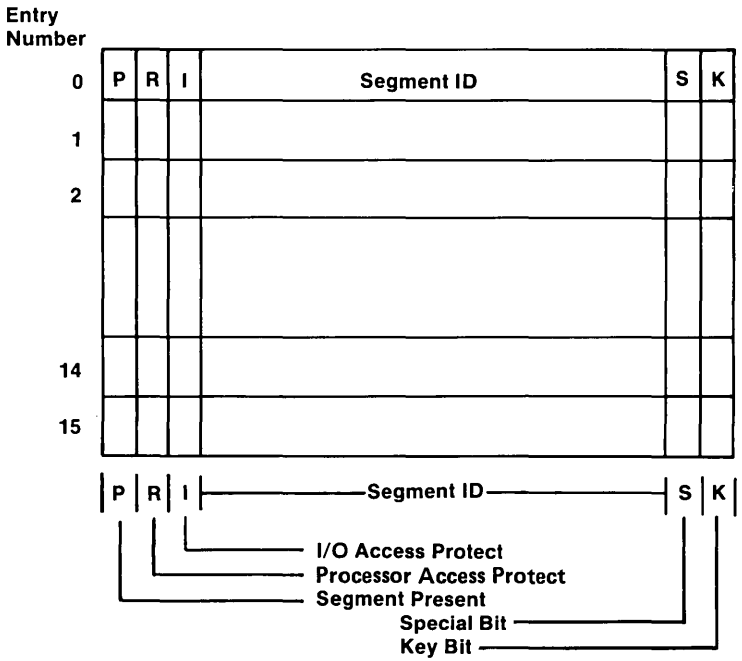


Figure 11-12. Segment Register Format

Generation of The Virtual Address

The 12-bit segment identifier is concatenated with bits 4 through 31 of the incoming effective address to form a 40-bit virtual address. The low-order 11 bits for 2K pages, or 12 bits for 4K pages, of the effective address are used as the byte address for the selected real page. These bits are not altered by the translation process. The remaining bits of the virtual address are then presented to the translation hardware. Figure 11-13 shows the generation of the virtual address using the segment identifier and the memory effective address.

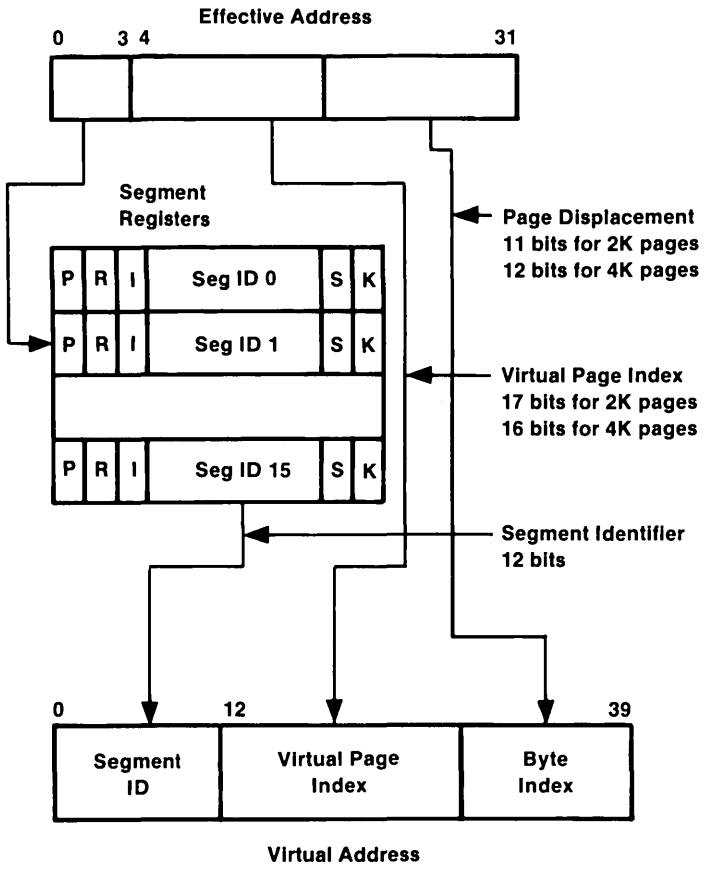


Figure 11-13. Generation of Virtual Address

TLB Operation

The memory management unit utilizes a translation look-aside buffer (TLB) to contain translations of the most recently used virtual addresses. Hardware is used to automatically update TLB entries from system memory page tables as new virtual addresses are presented to the TLBs for translation. A simplified data-flow of the translation hardware is shown in Figure 11-14 on page 11-100 and the format of each TLB is shown Figure 11-15 on page 11-101.

The memory management unit utilizes two TLBs with 16 entries per TLB (2-way set associative with 16 congruence classes). The low-order 4 bits of the virtual page index are used to address both TLBs in parallel. The address tag entry in each TLB is compared with the segment identifier concatenated with the remaining bits of the virtual page index (25 bits for 2K pages, or 24 bits for 4K pages). If either of the two compares are equal and the TLB entry is valid (as indicated by the valid bit), the associated TLB contains the translation information for the given virtual address.

The real page number field in the selected TLB entry contains the real page number in system memory that is mapped to the given virtual address. If this is not a special segment, the access is checked for memory protect violations using the key bits from the TLB entry and the key bit from the segment register before the access is allowed. If this is a special segment, as indicated by the special bit in the segment register, lockbit processing is performed before the access is allowed. The memory protect facility is described in "Memory Protection Processing" on page 11-109; and special segment processing is described "Lockbit Processing" on page 11-110. If the access is permitted, system memory is then accessed, and the reference and change bits associated with the page are updated. "Reference And Change Bits" on page 11-111 describes the setting of the reference and change bits.

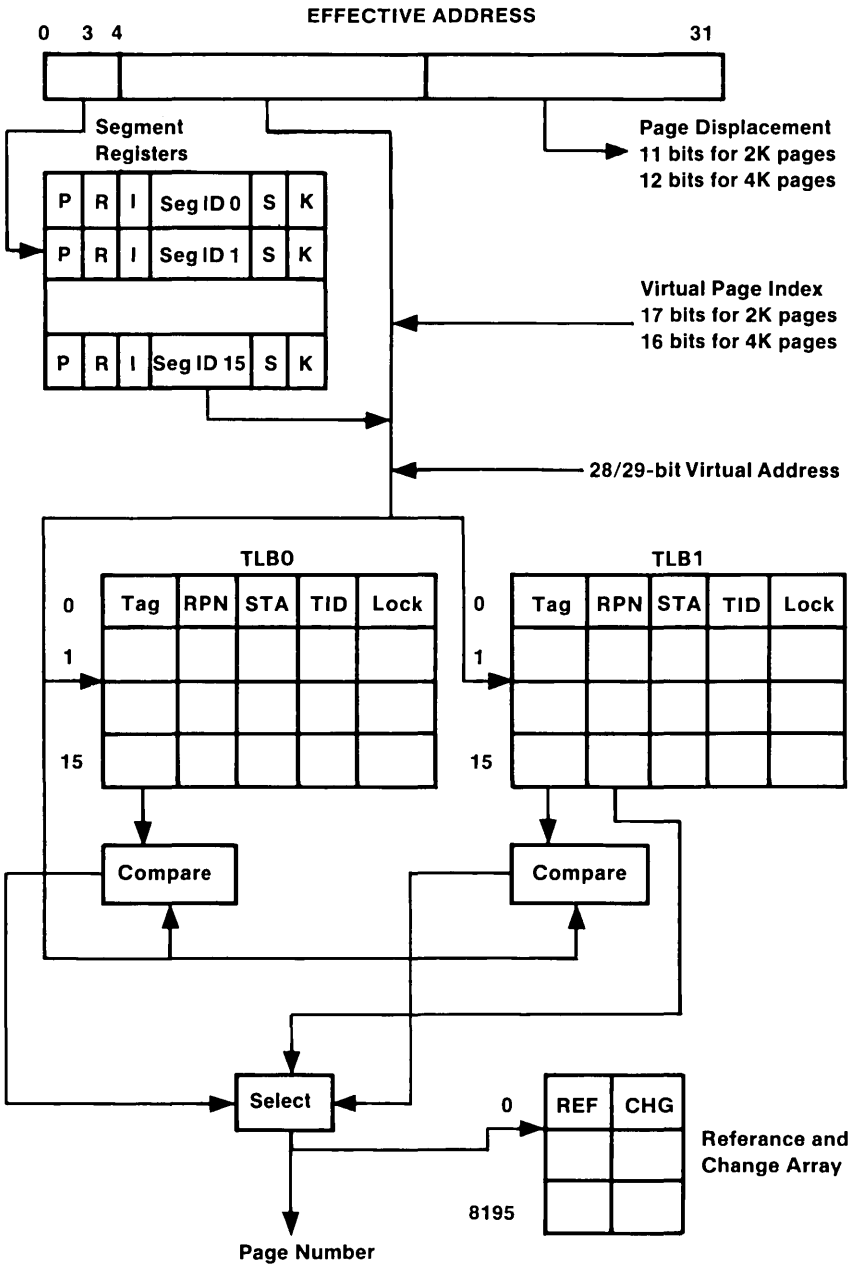


Figure 11-14. Address Translation Mechanism

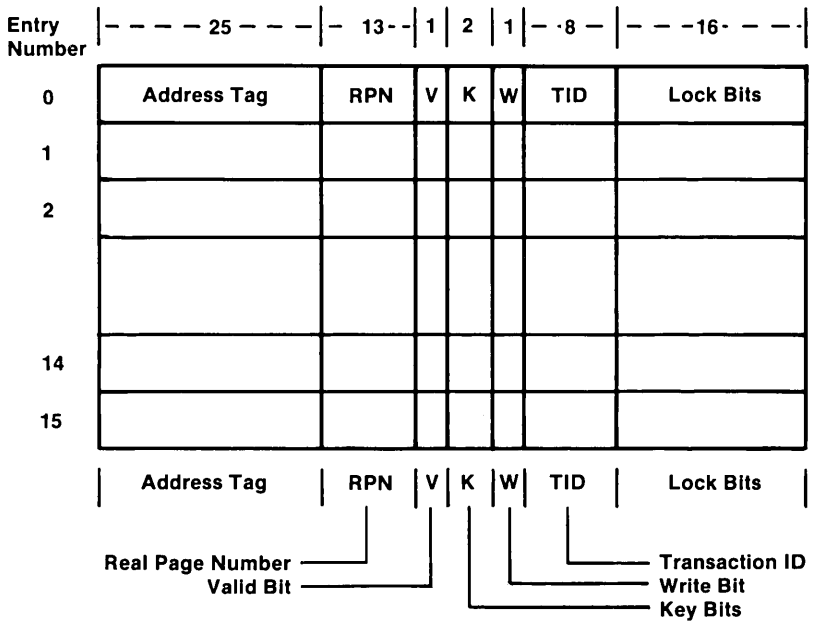


Figure 11-15. TLB Contents. One of two TLBs shown.

TLB Reload from System Memory Page Tables

If a match is not obtained from the two TLB compares, the address translation logic attempts to reload the faulting TLB entry from the page table entries in system memory. The system memory page table is resident in real memory and logically consists of two parts, a hash anchor table (HAT) and an inverted page table (IPT). The HAT maps any virtual address to any real page through a hashing function. The IPT defines which virtual address is currently assigned to each real page. There is one HAT and one IPT entry for each page of real memory.

Translation of a virtual address to a real address is accomplished by first exclusive 'ORing' selected low-order bits of the effective address with bits from the segment identifier. This hashed address is then used to index into the HAT. The selected HAT entry points to the beginning of a list of IPT entries to be searched for the given virtual address. Entries in the list of IPT entries to be searched are linked by a pointer in each entry that points to the next IPT entry to be searched. A flag bit in the IPT entry indicates the end of the search chain. Since the hashing function can produce the same HAT address for several different effective addresses, several virtual address entries in the IPT chain must be searched.

HAT and IPT Format

For hardware efficiency reasons, the HAT and IPT are combined into one structure that can be addressed by one indexing structure. There is one entry in the combined HAT and IPT for each page of real memory. For example, 1 megabyte of memory organized as 2K-byte pages requires 512 entries and 512K-bytes organized as 4K-pages requires 128 entries. The format of the combined HAT and IPT entries is shown in Figure 11-16. The HAT/IPT contains 16 bytes for each entry and starts on an address that is a multiple of the table size.

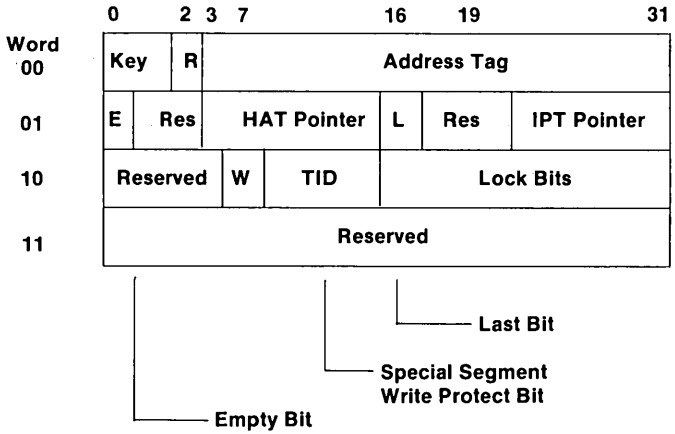


Figure 11-16. System Memory Inverted Page Table Entry Format

The first word in each entry contains the address tag which is composed of the segment identifier || virtual page index. Note that for 2K pages the address tag is 29 bits, and for 4K pages it is 28 bits. If a page size of 4K is used, the 28-bit address tag is stored in bits 3 through 30. Bit 2 is reserved. The first word also contains a 2-bit key that is used for memory protection as described in "Memory Protection Processing" on page 11-109.

The second word contains the HAT pointer, IPT pointer, and valid bits for each pointer. Use of the pointers is described in "HAT Address Generation" on page 11-104 and "IPT Search" on page 11-105.

The third word contains the write protect, lock bits, and TID for special segments. Use of these fields are described in "Memory Protection Processing" on page 11-109.

The fourth word is not used for TLB reloading, and is reserved for future use.

HAT/IPT Base Address

The HAT/IPT base address is the starting address for the system memory inverted page table (combined HAT/IPT), and is computed based on a field in the translation control register (TCR), the memory size defined by the RAM configuration register, and the page size defined in the translation control register. The value contained in the HAT/IPT base address field in the TCR is multiplied by value shown in Figure 11-17 to get the starting address of the HAT/IPT. Also shown in Figure 11-17 is the size of the HAT/IPT for each memory size and page size.

Memory Size Bytes	Page Size Bytes	HAT/IPT Size (Entries/Bytes	HAT/IPT Base Address Multiplier
64K	2K	32/512	512
64K	4K	16/256	256
128K	2K	64/51K	1024
128K	4K	32/512	512
256K	2K	120/2K	2048
256K	4K	64/1K	1024
512K	2K	256/4K	4096
512K	4K	128/2K	2048
1M	2K	512/8K	8192
1M	4K	256/4K	4096
2M	2K	1024/16K	16284
2M	4K	512/8K	8192
4M	2K	1048/32K	32768
4M	4K	1024/16K	16284
8M	2K	4096/64K	65536
8M	4K	2048/32K	32768
16M	2K	8192/128K	131072

Figure 11-17 (Part 1 of 2). HAT/IPT Base Address Multiplier

Memory Size Bytes	Page Size Bytes	HAT/IPT Size (Entries/Bytes	HAT/IPT Base Address Multiplier
16M	4K	4096/64K	65536

Figure 11-17 (Part 2 of 2). HAT/IPT Base Address Multiplier

HAT Address Generation

The HAT index is computed by exclusive 'ORing' selected bits from the segment identifier with bits from the effective address. The number of bits used is chosen so that the resulting index selects one of the entries in the HAT/IPT. The bits used for generation of the HAT Index are listed in Figure 11-18. The memory address of the selected HAT entry is computed as: HAT/IPT Base Address + HAT Index | | 0100.

The selected HAT entry is accessed and the empty bit is checked to determine if the IPT search chain is empty. If the empty bit is 1, there is no page mapped to the given virtual address and a page fault is reported as described in "Memory Exception Register" on page 11-119. If the empty bit is 0, entries in the IPT search chain exist, and they are searched. The HAT pointer field of the selected HAT entry is then used as a pointer to the start of the IPT search chain.

Memory Size (Bytes)	Page Size (Bytes)	Number of Pages	Segment ID Bits	Effective Address Bits	Index (Number of Bits)
64K	2K	32	7 - 11	16 - 20	5
64K	4K	16	8 - 11	16 - 19	4
128K	2K	64	6 - 11	15 - 20	6
128K	4K	32	7 - 11	15 - 19	5
256K	2K	128	5 - 11	14 - 20	7
256K	4K	64	6 - 11	14 - 19	6
512K	2K	256	4 - 11	13 - 20	8

Figure 11-18 (Part 1 of 2). HAT/IPT Index Generation Source Fields

Memory Size (Bytes)	Page Size (Bytes)	Number of Pages	Segment ID Bits	Effective Address Bits	Index (Number of Bits)
512K	4K	128	5 - 11	13 - 19	7
1M	2K	512	3 - 11	12 - 20	9
1M	4K	256	4 - 11	12 - 19	8
2M	2K	1024	2 - 11	11 - 20	10
2M	4K	512	3 - 11	11 - 19	9
4M	2K	2048	1 - 11	10 - 20	11
4M	4K	1024	2 - 11	10 - 19	10
8M	2K	4096	0 - 11	9 - 20	12
8M	4K	2048	1 - 11	9 - 19	11
16M	2K	8192	0 - 11	8 - 20	13
16M	4K	4096	0 - 11	8 - 19	12

Figure 11-18 (Part 2 of 2). HAT/IPT Index Generation Source Fields

IPT Search

The HAT pointer previously accessed is used as the starting index into the IPT. The memory address of the first IPT entry is computed as: HAT/IPT Base Address + HAT Pointer | | 0000.

The first entry in the IPT is accessed and the address tag is compared to the given virtual address. If the two are equal, the real page assigned to the virtual address has been located, and the faulting TLB entry can be reloaded. Reloading of the TLB entry is described in "TLB Reload from System Memory Page Tables" on page 11-101. If the two are not equal, the IPT search continues by accessing the IPT pointer. The IPT pointer address is computed as: HAT/IPT base address + HAT pointer | | 0100. The IPT pointer is then accessed, and the last bit is checked to determine if additional entries in the IPT search chain exist. If the last bit is a 0, there are additional entries and the search process continues. If the last bit is a 1, there are no additional IPT entries to be searched, and a page fault is reported as described in "Memory Exception Register" on page 11-119.

If there are additional IPT entries to be searched, the address of the next IPT entry for searching is computed as: HAT/IPT Base Address + IPT Pointer | | 0000. This address accesses the next entry in the IPT search chain; the address tag contained in the selected entry is compared to the

given virtual address. If the two are equal, the real page assigned to the virtual address has been located and the faulting TLB entry can be reloaded. Reloading of the TLB entry is described in "TLB Reload from System Memory Page Tables" on page 11-101. If the two are not equal, the search process continues by accessing the pointer to the next entry to be searched. The address of the pointer to the next entry is computed as: HAT/IPT base address + IPT pointer | | 0100. This word is then accessed, and the last bit is checked to determine if there are additional entries in the IPT search chain. If the last bit is a 1, there are no additional IPT entries to be searched, and a page fault is reported as described in "Memory Exception Register" on page 11-119. If the last bit is a 0, there are additional entries and the search process continues. The current IPT pointer is used to access subsequent entries using the previously described process, until either the address tag in the IPT entry is equal to the given virtual address, or no match is found and the last bit indicates no further entries exist in the search chain.

TLB Reload

If an IPT entry is found with an address tag field equal to the given virtual address, the faulting TLB entry is reloaded. Reloading consists of selecting the least recently used TLB entry for the congruence class of the faulting virtual address and then loading the selected entry with the given virtual address tag field, the corresponding real page number and the key bits. If this is a special segment as indicated by the special bit in the segment register, then the write bit, TID, and lock bits are also reloaded.

Hardware is used to determine the least recently used TLB entry in each congruence class. Since the low-order bits of the virtual address determine the congruence class, the only decision to be made is which TLB should have the selected entry replaced. One of the two TLBs is then selected according to which TLB contained the entry in the given congruence class that was referenced the furthest in the past.

Once the least recently used TLB entry for the given congruence class is determined, the selected TLB entry can be reloaded. The address tag field and key bits are reloaded from the IPT entry contained in system memory. The address of this entry was previously computed in the IPT search process. Since the IPT index computed in the search process is equal to the real page number, this value is used to reload the real page number field in the TLB. If this is a special segment, as indicated by the special bit in the segment register, the write bit, TID, and lock bits are also reloaded. The write bit, TID, and lock bits are reloaded by accessing the third word in the selected IPT entry.

TLB Reload Performance

The time required to reload a TLB entry is a function of the IPT search chain length, and whether the TLB entry is for a special or nonspecial segment. The TLB reload time can be computed from the sum of the following components.

- Base time
- IPT search time
- Special segment time
- Page fault time
- Correctable ECC error time.

The base time is approximately equal to 850 nanoseconds.

IPT search time is a function of the IPT search chain length. If the empty bit in the HAT pointer indicates there are no entries in the IPT, the IPT search time is 0. If the empty bit in the HAT pointer indicates there are valid entries in the IPT, the IPT search time can be computed as 510 nanoseconds times the number of entries searched in the IPT.

Special segment time is added only for special segments, and is equal to 340 nanoseconds.

Page fault time is added only if a page fault condition is detected during the IPT search. The page fault time is equal to 170 nanoseconds.

Correctable ECC error time is added for each correctable ECC error, which occurs during the TLB reload. Each correctable ECC error requires 170 nanoseconds for correction.

For example, consider the calculation of the TLB reload time for special segment which is successful (that is, no page fault), and a search of two IPT entries. Also, assume no correctable ECC errors occur during the reload. This results in a total TLB reload time of 850 nsec. + (510 nsec. times 2) + 340 nsec., or 2.2 microseconds.

Memory Access Control

The memory management unit provides three access control facilities. The first facility determines if an access is allowed, based on the source of the request (either system processor or an I/O device) and the system processor access protect and I/O access protect bits in the selected segment register. After an access has been validated on a segment basis, there is a second facility that applies to nonspecial segments which provides read/write protection for each page of real memory. A third facility applies only to special segments and is used to support the PL.8 Persistent data type. These access control facilities apply only to translated requests. Memory protection and lockbit processing are disabled for virtual equal to real mode. Virtual equal to real mode is enabled by the segment register zero virtual equal to real bit (bit 16) in the translation control register. When this bit is set to 1, translated accesses to segment register zero are treated as virtual equal to real. If a violation is detected by any of these facilities, the memory access is terminated and an exception reported to the system processor as described in "Memory Exception Register" on page 11-119.

Segment Protection Processing

Segment protection processing applies to all translated requests, including requests made in virtual equal real mode. The high-order 4 bits of the incoming effective address are used to index into the segment table to select one of 16 segment registers. A system processor access protection bit and an I/O access protection bit are obtained from the selected segment register. These bits, in conjunction with the source of the request (either system processor or an I/O device), determine whether the requested access is permitted for the selected segment. The source determination of the request is made by the tag field on the processor channel associated with each request. If the tag indicates the request is from the system processor, the system processor access protect bit determines if the access is valid. If the tag is from any device other than the system processor, (such as the I/O channel controller) the I/O access protect bit determines if the access is valid. Access is controlled as shown in Figure 11-19.

Access Type	System Processor Access Protect Bit	I/O Access Protect Bit	Access Permitted
system processor	0	–	Yes
system processor	1	–	No
I/O	–	0	Yes
I/O	–	1	No

Figure 11-19. Segment Protection Processing

If the access is not allowed, then the translation is terminated, and a segment protection exception is reported to the system processor as described in "Memory Exception Register" on page 11-119.

Memory Protection Processing

Memory protection processing applies only to virtual requests to nonspecial segments. Memory protection processing is disabled for virtual equal real accesses. Once a correspondence between a virtual and a real address is determined by the TLB, the requested access is verified to insure proper access authority. This facility allows each page to be marked as no access, read only, or read/write.

Access control is a function of the 1-bit protection key in the selected segment register, the 2-bit key in the TLB entry, and whether the access is a load or store operation. For memory protection processing, a test-and-set (TSH) operation is treated as a store. Access is controlled as shown in Figure 11-20 on page 11-110.

Key In TLB	Type of Page	Protect Key In Seg Reg	Access Permitted Load	Store
00	Key 0 fetch-protected	0	Yes	Yes
		1	No	No
01	Key 0 read/write	0	Yes	Yes
		1	Yes	No
10	Public read/write	0	Yes	Yes
		1	Yes	Yes
11	Public read-only	0	Yes	No
		1	Yes	No

Figure 11-20. Protection Key Processing

If the access is not allowed, then the translation is terminated, and a protection exception is reported to the system processor as described in "Memory Exception Register" on page 11-119.

Lockbit Processing

Lockbit processing applies only to virtual requests to special segments as indicated by the special bit in the selected segment register. Lockbit processing is disabled for virtual equal real accesses. Lockbit processing allows the operating system to automatically monitor changes to persistent variables and to journal changes, create shadow pages, and perform other processing required for data base consistency. Lockbits also extend the protection from the page size resolution (either 2K or 4K-bytes) provided by the memory protect facility to lines of either 128 or 256 bytes. A resolution of 128 bytes is provided for 2K pages, and 256 bytes for 4K pages. The individual line lockbit is selected by bits (21-24) of the effective address for 2K pages, and bits (20-23) for 4K pages. The proper lockbit is selected by these four bits with a value of 0000 selecting lockbit zero (bit 16 in the lockbit field in the HAT/IPT entry), and a value of 0001 selecting lockbit one (bit 17 in the lockbit field in the HAT/IPT entry). The selection of the proper lockbit continues in this manner with a value of 1111 selecting lockbit fifteen (bit 31 in the lockbit field in the HAT/IPT entry).

Access control is a function of the 1-bit write key in the selected TLB entry, the lockbit value of the selected line, the TID compare, and whether the access is a load or store operation. For lockbit processing, a test-and-set (TSH) operation is treated as a store. Access is controlled as shown in Figure 11-21.

Current TID Compared to TID In TLB	Write Bit In TLB	Lockbit Value For Selected Line	Access Permitted	
			Load	Store
Equal	1	1	Yes	Yes
		0	Yes	No
Equal	0	1	Yes	No
		0	No	No
Not Equal	—	—	No	No

Figure 11-21. Lockbit Processing

The data memory exception reports a lockbit violation. This violation may not represent an error; it may be simply an indication that a newly modified line must be processed by the operating system.

Reference And Change Bits

Reference and change bits are provided for each page of real memory. These bits are in arrays external to the memory management unit and are updated, as required, for each memory access. The reference bit is set to 1 if the corresponding real page is accessed for either a read or write operation. The change bit is set if the corresponding page is written.

Reference and change bits are accessible via I/O read and write instructions (IOR and IOW) from the system processor. Reference and change bits for each page of real memory start at the I/O address specified by the I/O Base Address Register plus X'1000'. The I/O address of the reference and change bits for a given page when 2K pages are used (TCR bit 23 set to zero) is given by the following equation.

I/O Address = Address Specified by I/O Base Address Register X'1000' Page Number

The I/O address of the reference and change bits for a given page when 4K pages are used (TCR bit 23 set to 1) is given by the following equation.

I/O Address = Address Specified by I/O Base Address Register X'1000'Page Number * 2

Each I/O address contains the reference and change bits for one page of real memory. The format of the reference and change bits is shown in Figure 11-22.

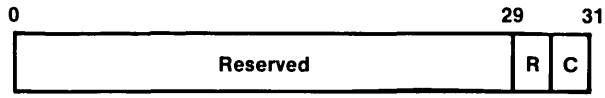


Figure 11-22. Reference and Change Bit Format

Data transferred by accesses to reference and change bits is defined as follows:

Bits 0-29 Reserved.

Bit 30 Reference Bit. Set to 1 when the corresponding real page is accessed for a read or write operation.

Bit 31 Change Bit. Set to 1 when the corresponding real page is accessed for a write operation.

Reference and change bits are not initialized by hardware. They are initialized and cleared by system software via IOW instructions. Since reference and change bits can be set or cleared by program execution, a write to clear or set reference and change bits followed by a read, will not necessarily read the same data that was written.

Translated accesses which do not have a corresponding TLB entry causes a superfluous memory access to the last page of the 16M-byte real memory address space. This superfluous access causes the reference bit for this page (page 8191 for 2K page size systems, and page 4095 for 4K page size systems) to be set. There are no other side effects from this superfluous access, other than setting of the reference bit for the last page.

Control Registers

There are 4 control registers used for defining the I/O base address, the RAM and ROM memory configuration, and the translation control information. There are 2 registers used to report errors detected by memory management unit, one register for use with the Compute Real Address function, one for the transaction identifier, and 16 segment registers. These registers are accessed by system software via I/O read and I/O write (IOR and IOW) instructions from the system processor. These registers are accessible only from privileged state. All control registers are initialized to 0 by the power-on-reset (POR) sequence.

I/O Base Address Register

The I/O base address register specifies which 64K block of I/O addresses are assigned to memory management unit. The I/O base address is equal to the value contained in the I/O base address register multiplied by 65536 (X'10000'). The format of the I/O base address register is shown in Figure 11-23.

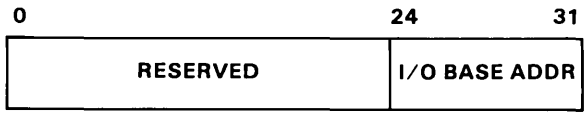


Figure 11-23. I/O Base Address Register

The I/O base address register is defined as follows:

Bits 0-23 Reserved.

Bits 24-31 I/O Base Address. This 8-bit value defines which 64K-byte block of I/O addresses are assigned to the memory management unit (these 8 bits are the most significant 8 bits in the I/O address recognized by memory management unit).

RAM Specification Register

The RAM specification register defines the RAM size and RAM starting address. The format for the RAM specification register is shown in Figure 11-24 on page 11-114.

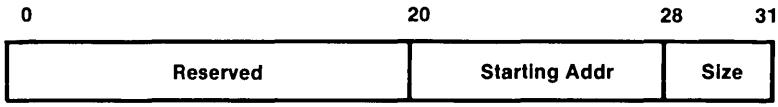


Figure 11-24. RAM Specification Register

The RAM specification register is defined as follows:

Bits 0-19 Reserved.

Bits 20-27 RAM Starting Address. This 8-bit field defines the starting address of RAM for nontranslated accesses. Translated accesses are unconditionally directed to RAM. For nontranslated accesses, the RAM starting address is used in conjunction with RAM Size to determine if an address is within the address range specified for memory management unit. The starting address of RAM is defined as a binary multiple of the RAM size, and is computed by multiplying the bits indicated below by the value specified by RAM Size.

RAM Size	Bits 20 21 22 23	Bits 24 25 26 27	Multiplier
64K	X X X X	X X X X	64K
128K	X X X X	X X X –	128K
256K	X X X X	X X – –	256K
512K	X X X X	X – – –	512K
1M	X X X X	– – – –	1M
2M	X X X –	– – – –	2M
4M	X X – –	– – – –	4M
8M	X – – –	– – – –	8M
16M	– – – –	– – – –	16M

X = bit used in address calculation
– = bit not used in address calculation

For example, if a memory size of 256K is specified, bits 20-25 specify which one of sixty four 256K-byte boundaries is the RAM starting address. If bits 20-25 are 011101, the RAM starting address is X'00740000'. If a RAM size of 1M-byte is specified, bits 20-23 specify which one of 16 1M-byte boundaries is the RAM starting address. If bits 20-23 are 1001, the RAM starting address is X'00900000'.

Note: To provide memory for the system processor PSWs, the RAM starting address must be set to zero.

Bits 28-31 RAM Size. This 4-bit field defines the size of RAM attached to memory management unit. RAM size is selectable from 64K-bytes to 16M-bytes as defined below.

Bits 28-31	RAM Size
0000	No RAM
0001 thru 0111	64K
1000	128K
1001	128K
1001	256K
1010	512K
1011	1M
1100	2M
1101	4M
1110	8M
1111	16M

ROM Specification Register

The ROM specification register defines the ROM starting address, ROM size, and whether parity is provided by ROM. ROM can be accessed only in non translated mode. The format of the ROM specification register is shown in Figure 11-25.

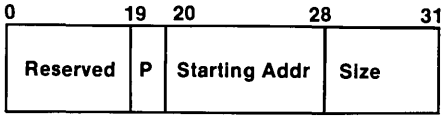


Figure 11-25. ROM Specification Register

The ROM specification register is defined as follows:

Bits 0-18 Reserved.

Bit 19 [P] ROM Parity Enable. This bit is used to enable internal memory management unit parity checking for ROM accesses. When this bit is set to 1, parity is checked by the memory management unit for ROM accesses (the ROM has parity bits). When set to 0, parity is not checked by the memory management unit for ROM

accesses. In both cases, the memory management unit generates parity for processor channel replies to ROM requests.

Note: Parity is not implemented on the processor board ROM, which requires that this bit always be set to 0.

Bits 20-27 ROM Starting Address. This eight-bit field defines the starting address of ROM for nontranslated accesses. For nontranslated accesses, the ROM starting address is used in conjunction with ROM size to determine if an address is within the address range specified for this memory controller. The starting address of ROM is defined as a binary multiple of the ROM size, and is computed by multiplying the bits indicated below by the value specified by ROM size.

Note: The processor board provides only a 64K-byte ROM.

ROM Size	Bits 20	21	22	23	Bits 24	25	26	27	Multiplier
64K	X	X	X	X	X	X	X	X	64K
128K	X	X	X	X	X	X	X	–	128K
256K	X	X	X	X	X	X	–	–	256K
512K	X	X	X	X	X	–	–	–	512K
1M	X	X	X	X	–	–	–	–	1M
2M	X	X	X	–	–	–	–	–	2M
4M	X	X	–	–	–	–	–	–	4M
8M	X	–	–	–	–	–	–	–	8M
16M	–	–	–	–	–	–	–	–	16M

X = bit used in address calculation
– = bit not used in address calculation

For example, if a ROM size of 64K is specified, bits 20-27 specify which one of 256 64K-byte boundaries is the ROM starting address. If bits 20-27 are 11001000, the ROM starting address is X'00C80000'.

Bits 28-31 ROM Size. This 4 bit field defines the size of ROM attached to the memory management unit. The processor board provides a 64K-byte ROM. ROM sizes of 128K thru 16M are not supported.

Note: The processor board provides a 64K-byte ROM. ROM sizes of 128K to 16M are not supported by the system processor board. The ROM size bits must be set to either no ROM or 64K-byte only.

Bits 28-31	ROM Size
0000	No ROM
0001 thru 0111	64K
1000	128K
1001	256K
1010	512K
1011	1M
1100	2M
1101	4M
1110	8M
1111	16M

Note: If the ROM address space, as defined by the ROM starting address and the ROM size, overlaps any of the RAM address space, as defined by the RAM starting address and the RAM size, the results are unpredictable. This means that if any ROM is used (bits 28 - 31 not zero), the maximum RAM size is 8M.

Translation Control Register

The translation control register (TCR) specifies:

- If segment register zero accesses are treated as virtual equal to real
- If interrupts are generated on a successful processor channel parity error retry
- If RAS diagnostic mode is enabled
- If unusually long IPT searches are terminated
- If interrupts are generated on correctable ECC errors
- If interrupts are generated on successful hardware TLB reload,
- The size of each page (either 2K or 4K-bytes)

- The starting address of the system memory page table (combined HAT and IPT).

The translation control register format is shown in Figure 11-26.

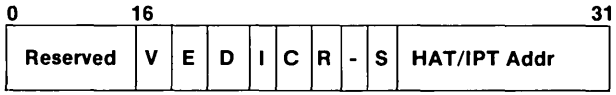


Figure 11-26. Translation Control Register

The translation control register is defined as follows:

Bits 0-15 Reserved.

Bit 16 (V) Segment Register Zero Virtual Equal To Real. When set to 1, all accesses to segment register zero are treated as a real (untranslated) address. In this mode, memory protection checking and lockbit processing are disabled, and the low-order 24 bits of the 32-bit virtual address are used as the memory address. Contents of the TLBs are not altered by virtual equal to real accesses. When segment register zero virtual equal to real is set to 0, normal protection processing and address translation applies to virtual accesses to segment register zero.

Bit 17 (E) Interrupt On Successful Parity Error Retry. When set to 1, this bit enables a machine check interrupt, unless RAS diagnostic mode is enabled, when a processor channel retry successfully completes a memory management unit reply that was previously unsuccessful due to detection of a parity error by the receiving device. A successful parity error retry sets the ACKD and NAKD bits in the MER (bits 15 and 16) to 1. When interrupt on successful parity error retry is set to 0 or RAS diagnostic mode is enabled (TCR bit 18 set to 1), no interrupt results from a successful retry.

Bit 18 (D) Enable RAS Diagnostic Mode. When set to 1, this bit enables ECC diagnostic checking. When set to 0, diagnostic mode is disabled. When RAS diagnostic mode is enabled, all machine check interrupts and exception replies (except addressing errors) from the memory management unit are disabled. Also, all store operations use the alternate check bits stored in the RMDR instead of the memory management unit generated ECC check bits for the stored data. See "RAS Mode Diagnostic Register" on page 11-132 for a complete description of RAS diagnostic mode.

Bit 19 (I) Terminate Long IPT Search. When set to 1, this bit terminates hardware searching of the IPT after 127 entries have been compared in a given search chain, and a matching entry is not found. When the IPT search is terminated, an exception reply is generated and the IPT specification error bit (bit 25) in the MER is set to 1. When terminate long IPT search is set to 0, the IPT search continues until a matching entry is found, or until the end of the search chain is detected.

Since each IPT search chain should contain only a few entries, use of this facility will normally report only software errors in constructing the search chains. Unusually long search chains, or search chains which contain an infinite loop are detected and reported by enabling this facility.

Bit 20 (C) Enable Interrupt On Correctable ECC Error. When set to 1, this bit enables reporting of correctable ECC errors via the machine check interrupt line to the system processor, unless RAS diagnostic mode is enabled. A correctable ECC error sets the correctable ECC error bit (bit 21) in the MER to one, and the MEAR is loaded with the memory address of the correctable error, if it is not locked. The MEAR is then locked to all subsequent real errors and no further ECC error machine check interrupts are reported until the RMDR is read. When enable interrupt on correctable ECC error is set to 0 or RAS diagnostic mode is enabled (TCR bit 18 set to 1), correctable ECC errors are not reported.

Bit 21 (R) Enable Interrupt On Successful TLB Reload. This bit enables reporting of successful hardware TLB reloading. When set to 1, a successful hardware TLB reload generates a exception reply, and sets the TLB reload bit (bit 22) in the MER to 1. When enable interrupt on successful TLB reload is set to 0, successful hardware reloading of TLB entries is not reported. This facility is used for software performance measurement of the TLBs.

Bit 22 Reserved.

Bit 23 (S) Page Size. A value of 0 is used for 2K-byte pages, and a value of 1 is used for 4K-byte pages.

Bits 24-31 HAT/IPT Base Address. This 8-bit field specifies the starting address of HAT/IPT entries in system memory. The value contained in this field is multiplied by a constant determined by the size of real memory and the page size, to determine the starting address of the HAT/IPT entries. For a page size of 2K-bytes, the base address is specified by bits 25-31, and for 4K-pages by bits 24-31. The constant for each memory size and page size configuration is listed in Figure 11-17 on page 11-103.

Memory Exception Register

The memory exception register (MER) reports errors in the translation process, system errors, and ECC errors for a memory access. Individual bits are provided to report each error condition detected by the memory management unit. In multiple errors, each error is reported by setting of the appropriate bit. Bits which were set by previous errors are not reset by subsequent errors.

The MER is initialized to 0 by the POR sequence. Once an exception is reported, system software is responsible for clearing the MER (via an IOW instruction) after the exception is processed.

The memory management unit provides a -PGM Check output to report suspected software errors. The - PGM Check output is activated when software related errors are detected. Reading the MER by an IOR instruction resets the -PGM CHECK output.

The memory exception register format is shown in Figure 11-27.

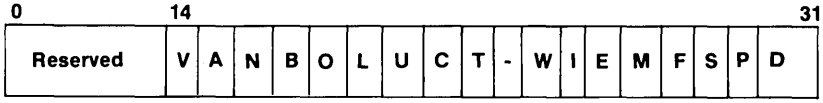


Figure 11-27. Memory Exception Register

The memory exception register is defined as follows:

Bits 0-13 Reserved.

Bit 14 (V) Segment Protection Violation. This bit is set to 1 when translation is terminated because segment protection processing determines that a memory access is not allowed. This bit is set only for protection errors detected during the processing of system processor generated requests.

Detection of a segment protection violation for any processor channel request generates a exception reply, unless RAS diagnostic mode is enabled. If the segment protection violation is caused by a system processor generated load or store, and the MEAR is not locked, the MEAR is loaded with the effective address causing the segment protection violation, and is set to the locked state.

Bit 15 (A) Processor channel ACKD. This bit is set to the value of the processor channel data cycle ACK signal when a reply from the memory management unit to another device on the processor channel is rejected. This bit, combined with the processor channel NAKD bit, allows software to determine the reason a reply was rejected. The four combinations of ACKD and NAKD are defined below:

ACKD	NAKD	Meaning
0	0	No Device Responded
0	1	Device Busy
1	0	Transfer Successful
1	1	Parity Error

Whenever a memory management unit reply is rejected, the memory management unit will retry the reply two times.

Programming Note: Setting Of ACKD And NAKD Bits

The processor channel ACKD and NAKD bits are loaded with the proper values whenever an processor channel retry occurs. This can occur during system operation, or when a support processor is used for system testing. In normal system operation, the ACKD and NAKD bits are loaded if there is no response to a memory management unit reply (ACKD=0, NAKD=0), a device responds busy to a memory management unit reply (ACKD=0, NAKD=1), or a device indicates a parity error to a memory management unit reply (ACKD=1, NAKD=1).

System software reads the value of the ACKD and NAKD bits to determine the reason a reply was rejected. Software is responsible for resetting these bits once they have been read.

Bit 16 (N) Processor channel NAKD. This bit is set to the value of the processor channel data cycle NAK signal when a reply from the memory management unit to another device on the processor channel is rejected. This bit, combined with the processor channel ACKD bit, allows software to determine the reason a reply was rejected. The four combinations of ACKD and NAKD are defined above.

Bit 17 (B) Invalid Memory Address. This bit is set to 1 when an address is within the address range specified for ROM or RAM by the ROM specification register and RAM specification register, but is not within the address range defined on the ROM and RAM memory boards attached to the memory management unit. For example, if the RAM specification register defines 4M-bytes of RAM starting at address 0, and only 3M-bytes of memory are actually attached to the memory management unit, an address from 0 to 4M will be accepted by the memory management unit, but addresses greater than 3M set this bit (invalid memory address). This bit is set for invalid memory addresses generated by any processor channel device.

Detection of an invalid memory address generates an exception reply for read accesses or translated write accesses, and activates the program check interrupt line for nontranslated write accesses. The MEAR is loaded with the invalid memory address, if it is not locked.

Bit 18 (O) Invalid I/O Address. This bit is set to 1 when an I/O operation to a reserved address within the 64K address range recognized by the memory management unit is detected, or when an I/O read to a write only address (such as a read to the address for purge TLB, purge TLB segment, purge TLB entry, or load real address) is detected.

Detection of an invalid I/O address generates an exception reply for I/O reads, unless RAS diagnostic mode is enabled, and activates the program check interrupt line for I/O writes. The MEAR is loaded with the invalid I/O address, if it is not locked.

Bit 19 (L) Access Type. This bit is set to 1 when an exception is caused by a load, and set to 0 when the exception is caused by a store or test-and-set. This bit is valid only for segment protection violation, IPT specification error, TLB specification error, page fault, protection, and data exceptions caused by a system processor load or store.

Bit 20 (U) Uncorrectable Memory Error. This bit is set to 1 when there is an uncorrectable ECC error.

Detection of an uncorrectable memory error generates an exception reply for read accesses or translated write accesses, and activates the machine check interrupt line, unless RAS diagnostic mode is enabled. The MEAR is loaded with the address of the uncorrectable memory error, if it is not locked. The MEAR is then locked to all subsequent real errors and ECC error machine checks are not generated until the RAS mode diagnostic register is read. The RMDR is loaded with ECC bits of the failing memory operation and locked until read.

Bit 21 (C) Correctable ECC Error. This bit is set to 1, unless the uncorrectable memory error bit is already set, when a correctable ECC error is detected during a memory operation for any processor channel device,

If interrupt on correctable ECC error is enabled (TCR bit 20 set to 1), and a correctable ECC error is detected, the machine check interrupt line is activated, unless RAS diagnostic mode is enabled. The MEAR is loaded with the memory address of the correctable ECC error, if it is not locked. The MEAR is then locked to all subsequent real errors and ECC error machine checks are not generated until the RAS mode diagnostic register is read. The RMDR is loaded with ECC bits of the failing memory operation and locked until read.

Bit 22 (T) Successful TLB Reload. This bit is set to 1 when interrupt on successful TLB reload is enabled (TCR bit 21 set to 1), and a TLB entry is successfully reloaded for any processor channel device.

If a successful TLB reload occurs and interrupt on successful TLB reload is enabled, an exception reply is generated, unless RAS diagnostic mode is enabled (such as successful TLB reload interrupts are disabled whenever RAS diagnostic mode, bit 18 in the TCR, is set to a 1).

Bit 23 Reserved.

Bit 24 (W) Write To ROM Attempted. This bit is set to 1 when an attempt to write to an address contained in the ROM address space is made by any processor channel device.

Detection of a write to ROM activates the program check interrupt line. The MEAR is loaded with the memory address causing the error, if it is not locked.

Bit 25 (I) IPT Specification Error. This bit is set to 1 when:

- Terminate long IPT search is enabled (TCR bit 19 set to 1)

- TLB miss causes an IPT search which fails to find a matching IPT entry
- The end of the IPT search chain after 127 IPT entries.

This bit is set only for IPT specification errors detected during the processing of system processor generated requests. Detection of an IPT specification error for any processor channel request generates an exception reply, unless RAS diagnostic mode is enabled. If the IPT specification error is caused by a system processor generated load or store, and the MEAR is not locked, the MEAR is loaded with the effective address causing the IPT specification error, and is set to the locked state.

Bit 26 (E) External Device Exception. This bit is set to 1 when an exception (segment protect, IPT specification error, page fault, protection, or data) is caused by the IOCC.

Bit 27 (M) Multiple Exception. This bit is set to 1 when more than one exception (segment protect, IPT specification error, page fault, protection, or data) has occurred before the exception indication has been cleared in the memory exception register and the MEAR is currently locked with a virtual error. This bit is set only if the multiple exception is caused by system processor loads or stores.

This bit normally indicates that system software has failed to process an exception. However, if an exception is caused by a Load Multiple (LM) or Store Multiple (STM) instruction, this bit can be set since the LM or STM instruction attempts to load or store all the registers specified in the instruction before the instruction is terminated due to an exception. If two virtual error bits are set, and the multiple exception bit is not set, one of the errors was caused by an unexecuted instruction prefetch.

Bit 28 (F) Page Fault. This bit is set to 1 when:

- Translation is terminated because no TLB entry or system memory page table entry contains the translation for a virtual address
- A TLB reload operation is in progress when a reset packet is received
- A TLB reload operation is terminated because an invalid address or an uncorrectable memory error is detected during a TLB reload operation.

This bit is set only for page faults detected during the processing of system processor generated requests. Detection of a page fault for a IOCC request generates a exception reply, unless RAS diagnostic mode is enabled. If the page fault is caused by a system processor load or store, and the MEAR is not locked, the MEAR is loaded with the effective address causing the page fault, and is set to the locked state.

Bit 29 (S) TLB Specification. This bit is set to 1 when translation is terminated because two TLB entries were found for the same virtual address for any processor channel request.

Detection of a TLB specification error generates a exception reply and activates the machine check interrupt line, unless RAS diagnostic mode is enabled. The MEAR is loaded with the effective address causing the TLB specification error, if it is not locked, and is set to the locked state.

If a TLB Specification error occurs during a store operation, the store occurs to an address that is the logical OR of the real page number value in the two matching TLB entries.

Bit 30 (P) Protection. This bit is set to 1 when translation is terminated because memory protection processing for a non-special segment determines that a memory access is not allowed. This bit is set only for protection errors detected during the processing of system processor generated requests.

Detection of a protection error for any processor channel request generates a exception reply, unless RAS diagnostic mode is enabled. If the protection error is caused by a system processor generated load or store, and the MEAR is not locked, the MEAR is loaded with the effective address causing the protection error, and is set to the locked state.

Bit 31 (D) Data. This bit is set to 1 when translation is terminated because transaction ID/lockbit processing for a special segment determines that a memory access is not allowed. This bit is set only for data errors detected during the processing of system processor generated requests.

Detection of a data error for any processor channel request generates a exception reply, unless RAS diagnostic mode is enabled. If the data error is caused by a system processor generated load or store, and the MEAR is not locked, the MEAR is loaded with the effective address causing the data error, and is set to the locked state.

Reading of the MER by Diagnostic Routines

Diagnostic routines must be aware that a cycle of delay exists between the occurrence of a memory error and the reporting of that error in the MER. If a load or store instruction which is expected to cause an error is followed immediately by an IOR of the MER, the value returned by the IOR may not have the corresponding error bits set. A load to register RX should be followed by a CAS RX, RX, R0 and then the IOR to guarantee that the correct MER value is read. If the faulting instruction is a store, insert an intervening load instruction before the IOR to assure that the correct MER value is returned.

Memory Exception Address Register

The memory exception address register (MEAR) contains the 32-bit effective address causing an exception reported by the memory exception register (MER) for an IPT specification error, page fault, TLB specification, segment protection violation, protection, or data exception, if the exception was caused by a system processor generated load or store operation. The MEAR is not loaded for exceptions caused by system processor instruction fetches, or by IOCC requests for the exceptions previously listed. System processor generated load and store operations cause the MEAR to be loaded with the 32-bit effective address causing the exception, if the MEAR is not locked. Detection of a TLB specification error for any virtual request causes the MEAR to be loaded with the 32-bit effective address causing the specification exception, if the MEAR is not locked. Once the MEAR is loaded with an effective address, it is set to the locked state.

In addition, the MEAR contains the 24-bit memory address causing an error reported by the MER for an invalid memory address, invalid I/O address, uncorrectable ECC error, correctable ECC error, or write to ROM attempted error. The MEAR is loaded when the system processor or any other processor channel devices causes one of these errors, and the MEAR is not locked. Loading of the MEAR with a memory address does not set the MEAR to the locked state. However, loading of the MEAR for all uncorrectable ECC errors and correctable ECC errors when interrupt on correctable ECC error or RAS diagnostic mode are enabled (bits 18 or 20 in the TCR set to one), prevents the MEAR from being loaded by other memory related errors until the RMDR is read by an IOR instruction. All virtual exceptions will still be loaded into the MEAR for system processor loads and stores, and the MEAR is then placed in the locked state.

The MEAR is initialized to the not locked state as part of the POR sequence. Writing of the MEAR as a result of an IPT specification error, page fault, TLB specification, segment protection violation, protection, or data exception causes the MEAR to be set to the locked state. Locking the MEAR prevents further exceptions from writing the MEAR. Reading or writing of the MEAR by software (via an IOR or IOW instruction) sets the MEAR to the not locked state.

In multiple errors (bit 27 of the MER set to one) the MEAR contains the address of the oldest exception. The memory exception address register format is shown in Figure 11-28.

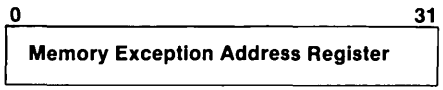


Figure 11-28. Memory Exception Address Register

The memory exception address register is defined as follows:

Bits 0-31 Memory Exception Address. The 32-bit effective memory address causing the exception reported by the MER for virtual accesses, or the 24-bit real memory address reported by the MER for real accesses. The 24-bit real memory address is contained in bits 8-31, and bits 0-7 are set to 0. In multiple errors (bit 27 of the

MER set to 1), the address contained in the MEAR is the address of the oldest exception.

Reading of the MEAR by Diagnostic Routines

Diagnostic routines must be aware that a cycle of delay exists between the occurrence of a memory error and the loading of the address of that error in the MEAR. If a load or store instruction which is expected to cause an error is followed immediately by an IOR of the MEAR, the value returned by the IOR may not have the corresponding error address. A load to register RX should be followed by a CAS RX,RX,R0 and then the IOR to guarantee that the correct MEAR value is read. If the faulting instruction is a store, insert an intervening load instruction before the IOR to assure that the correct MEAR value is returned.

Translated Real Address Register

The translated real address register (TRAR) contains the real memory address determined by the Compute Real Address operation. The Compute Real Address function is used to determine if a virtual address is currently mapped in real memory, and the corresponding real address if the virtual address is mapped. The Compute Real Address function is described in "Compute Real Address" on page 11-135. The translated real address register format is shown in Figure 11-29.

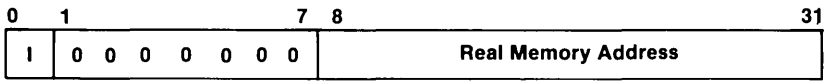


Figure 11-29. Translated Real Address Register

The translated real address register is defined as follows:

Bit 0 (I) — Invalid Bit. This bit is set to 1 if the translation failed, and is set to 0 if the translation is successful. Segment protection, memory protect, and lockbit checking are not included in determination of the invalid bit.

Bits 1-7 — Zeros. This 7-bit field is always 0.

Bits 8-31 — Real Memory Address. This 24-bit field contains the real memory address mapped to the given virtual address if translation was successful. This field is unpredictable if translation fails.

Transaction Identifier Register

The transaction identifier register (TID) contains the eight-bit identifier of the task currently defined as the owner of special segments. If a segment is defined as a special segment by the special bit in the selected segment register, then lockbit processing as described in "Lockbit Processing" on page 11-110 applies to the memory access. Lockbit processing uses the value contained in the TID and compares it against the TID entry in the TLB to determine if the memory access is permitted. The format of the transaction identifier register is shown in Figure 11-30.

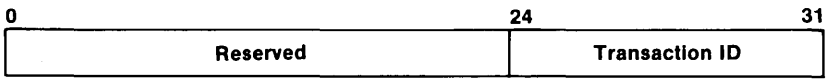


Figure 11-30. Transaction Identifier Register

The transaction identifier register is defined as follows:

Bits 0-23 Reserved.

Bits 24-31 Transaction Identifier. This 8-bit value specifies the owner of special segments.

Segment Registers

The sixteen segment registers contain the segment present bit, the system processor access protection bit, the I/O access protection bit, the segment identifier, special bit, and key bit. The segment present bit indicates whether the corresponding segment is assigned to the memory management unit. The system processor and I/O access protect bits indicate whether system processor or I/O accesses are permitted to the corresponding segment. The 12-bit Segment Identifier specifies one of 4096 256M-byte virtual memory segments. The special bit indicates this is a special segment and lockbit processing applies. The key bit indicates the level of access authority associated with the currently executing task with respect to memory accesses within the given segment. The format of each segment register is shown in Figure 11-31.

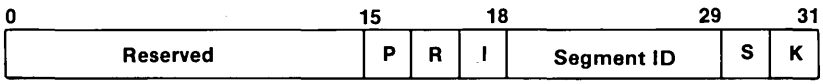


Figure 11-31. Segment Register (One Of Sixteen)

The content of each segment register is defined as follows:

Bits 0-14	Reserved.
Bit 15 (P)	Segment Present. This bit is set to 1 if the corresponding segment is assigned to the memory management unit. A virtual request will only be accepted by the memory management unit if the selected segment is assigned to the memory management unit. Virtual requests which select a segment that is not assigned to the memory management unit (segment present bit set to 0) are not accepted. This facility allows multiple memory management units (or other processor channel devices) to accept only selected virtual addresses.
Bit 16 (R)	System Processor Access Protect. This bit is set to 0 if system processor accesses are allowed for the given segment, and set to 1 if system processor accesses are not allowed for the given segment. Use of this bit for segment protection is described in "Segment Protection Processing" on page 11-108.
Bit 17 (I)	I/O Access Protect. This bit is set to 0 if I/O (non system processor) accesses are allowed for the given segment, and set to 1 if I/O (non system processor) accesses are not allowed for the given segment. Use of this bit for segment protection is described in "Segment Protection Processing" on page 11-108.
Bits 18-29	Segment Identifier. This 12-bit quantity specifies one of 4096 256M-byte virtual memory segments.
Bit 30 (S)	Special Bit. This bit is set to 1 for special segments, and set to 0 for non special segments.
Bit 31 (K)	Key Bit. This bit determines the level of access authority of the currently executing task for accesses within the given segment. Use of this bit for memory access control is described in "Memory Protection Processing" on page 11-109.

TLB Entries

Each of the two TLBs contain sixteen entries, which provide the necessary translation and control information for the conversion of a virtual address to a real address. In addition, each TLB entry contains information used for memory access control. Since the TLB contents are automatically updated from the system memory page table by hardware, writing of a TLB entry followed by a read will not necessarily read the same data which was written. Also, altering TLB entries can cause unpredictable results since the correspondence between virtual and real addresses will be destroyed. Access to the TLB contents is provided for diagnostic purposes only, and should only be made in non-translated mode. A write to a TLB entry in non-translated mode with all other translated accesses disabled, followed by a read, will read the same data that was written.

Each TLB entry is logically a 66-bit quantity composed of:

- A 25-bit address tag
- A 13-bit real page number

- A valid bit
- A 2-bit key
- A write bit
- A 8-bit transaction ID
- And 16 lockbits.

Each TLB entry is partitioned into three fields which are individually addressable. The format for each of the TLB fields are described in "TLB Address Tag" on page 11-129 through "TLB Write Bit, Transaction ID, and Lockbits" on page 11-131.

TLB Address Tag

The TLB Address Tag field contains the high-order 25 bits of the segment identifier | | virtual page index for 2K pages, and the high-order 24 bits for 4K pages. The format of the address tag field for each TLB entry is shown in Figure 11-32.

Figure 11-32. TLB Address Tag Field (One Of Sixteen Per TLB)

The content of each TLB address tag field is defined as follows:

Bits 0-2 Reserved.

Bits 3-27 Address Tag. This field contains the high-order 25 bits of the segment identifier | | virtual page index for 2K pages, and the high-order 24 bits for 4K pages. For 4K pages, the address tag is contained in bits (3-26).

Bits 28-31 Reserved.

TLB Real Page Number, Valid Bit, and Key Bits

This field contains the real page number assigned to the virtual address contained in the address tag field of the TLB entry. This field also includes a valid bit to indicate the given TLB entry contains valid information, and key bits for the access authority required for a given page. The format of this field for each TLB entry is shown in Figure 11-33.

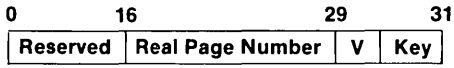


Figure 11-33. TLB Real Page Number, Valid, and Key Bits (One of Sixteen Per TLB)

The content of the real page number, valid, and key Bits field is defined as follows:

Bits 0-15 Reserved.

Bits 16-28 Real Page Number (RPN). This 13-bit field specifies one of 8192 real pages. If less than 8192 pages are implemented, only those low-order bits required to address the number of implemented pages are used. The RPN is contained in bits (16-27) for 4K pages, and in bits (16-28) for 2K pages. Bit 28 is 0 for 4k pages.

Bit 29 (V) Valid Bit. This bit is 1 when the selected TLB entry contains valid information. This bit is 0 if the TLB entry contains invalid information.

Bits 30-31 Key Bits. This 2-bit field defines the access authority for each page. Use of the key bits are described in "Memory Protection Processing" on page 11-109.

TLB Write Bit, Transaction ID, and Lockbits

This field contains the write bit, transaction ID, and lockbits assigned to the virtual address contained in the address tag field of the TLB entry, if the TLB entry is for a special segment. The format of this field for each TLB entry is shown in Figure 11-34.

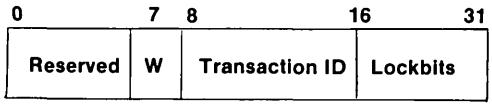


Figure 11-34. Write Bit, Transaction ID, and Lockbits (One of Sixteen Per TLB)

The content of each TLB write bit, transaction ID, and lockbit field is defined as follows:

Bits 0-6 Reserved.

Bit 7 (W) Write Bit. This bit defines the access authority associated with each page for special segments. Use of this bit in lockbit processing is described in "Memory Protection Processing" on page 11-109.

Bits 8-15 Transaction Identifier. This 8-bit field defines the task which currently owns the selected page within a special segment. Use of these bits in lockbit processing are described in "Memory Protection Processing" on page 11-109.

Bits 16-31 Lockbits. This 16-bit field defines the access authority for each line within a 2K or 4K page for special segments. A line is 128 bytes for 2K pages, and 256 bytes for 4K pages. Use of these bits in lockbit processing are described in "Memory Protection Processing" on page 11-109.

RAS Mode Diagnostic Register

The RAS mode diagnostic register (RMDR) is used to verify operation of the ECC logic, and to identify failing bits in memory. The RMDR allows systems software to determine which memory bits are failing by examining the check bits read from the location causing the ECC error. Alternate ECC check bits can be supplied from a field in the RMDR to verify proper operation of the ECC logic in the memory management unit. The format of the RMDR is shown in Figure 11-35.

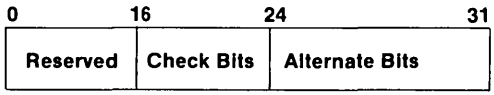


Figure 11-35. RAS Mode Diagnostic Register

The RAS mode diagnostic register is defined as follows:

Bits 0-15 Reserved.

Bits 16-23 Array Check Bits. This 8-bit field is loaded with the eight ECC check bits read from RAM for the first load or read-modify-write store (such as store character, store half, or test-and-set) when a correctable or uncorrectable ECC error is detected and RAS diagnostic mode or interrupt on correctable ECC error are enabled (bits 18 or 20 of the TCR set to 1). If RAS diagnostic mode and interrupt on correctable ECC error are disabled (bits 18 and 20 of the TCR set to 0), this field will be loaded only by uncorrectable ECC errors. Once this field is loaded, the value in the RMDR is locked, and is not altered until the RMDR is read by system software via an IOR instruction. The MEAR remains locked to all memory related errors until the RMDR is read. The eight array check bits are read only, an I/O write to these bits will not alter their value. The correspondence of the eight array check bits to the eight ECC check bits described in "ECC Checking" on page 11-141 is defined below:

RMDR Bit	ECC Check Bit
Bit 16	ECC0
Bit 17	ECC1
Bit 18	ECC2
Bit 19	ECC3
Bit 20	ECC4
Bit 21	ECC5
Bit 22	ECC6
Bit 23	ECC7

Bits 24-31 Alternate Check Bits. This 8-bit field contains alternate ECC check bits that are written to RAM for store accesses when RAS diagnostic mode is enabled (bit 18 of the TCR set to 1). This field allows software to supply a source of ECC check bits for diagnostic or error recovery procedures. When RAS diagnostic mode is enabled, the alternate check bits are supplied to the RAM as ECC check bits in place of the ECC check bits which are normally generated by internal memory management unit hardware. When RAS diagnostic mode is disabled (bit 18 of the TCR set to 0), check bits generated by the internal memory management unit ECC logic are supplied to RAM. The correspondence of bits in this field to ECC check bits are defined below:

RMDR Bit	**ECC Check Bit**
Bit 24	ECC0
Bit 25	ECC1
Bit 26	ECC2
Bit 27	ECC3
Bit 28	ECC4
Bit 29	ECC5
Bit 30	ECC6
Bit 31	ECC7

Translation Assist Functions

The memory management unit provides hardware support for frequently required translation functions. This hardware provides the ability to selectively invalidate TLB entries, and to perform a "compute real address" function.

Purging TLB Entries

As changes to the virtual-to-real address mapping are made, it is necessary for system software to synchronize the contents of the TLBs with the contents of the page table in system memory. Entries must be purged (invalidated) to ensure that obsolete mapping information is not used for a subsequent translation.

The memory management unit provides three functions to assist in the synchronization of TLB entries with the contents of the page table in system memory. These functions can be used to invalidate the entire TLB contents, or to invalidate only selected TLB entries. These functions are invoked by I/O write instructions (IOW) directed to specific I/O addresses within the 64K-byte block of I/O addresses recognized by the memory management unit. The address assignment for each of these functions is given in "I/O Address Assignments" on page 11-136.

Invalidate Entire TLB

This function causes all TLB entries to be invalidated. This will force the TLB contents to be updated from page tables in system memory for subsequent translations.

An I/O write to the address associated with this function causes all TLB entries to be invalidated. The data transferred by the I/O write instruction is not used.

Invalidate TLB Entries In Specified Segment

This function causes all TLB entries with the specified segment identifier to be invalidated. Subsequent translations using this segment identifier causes the TLB contents to be updated from page tables in system memory.

An I/O write to the address associated with this function causes TLB entries with the specified segment identifier to be invalidated. Bits (0-3) of the data transferred by the I/O write instruction are used to select the segment identifier. All TLB entries containing this segment identifier are invalidated. Subsequent translations with an effective address within the invalidated segment causes the TLB contents to be updated from the page table in system memory.

Invalidate TLB Entry For Specified Effective Address

This function causes the TLB entry with the specified effective address to be invalidated. Subsequent translations with an effective address within the page containing the specified effective address causes the TLB contents to be updated from the page table in system memory.

An I/O write to the address associated with this function causes the TLB entry with the specified effective address to be invalidated. Bits (0-31) of the data transferred by the I/O write instruction are used as the effective address. The normal translation process is applied using the segment register contents contained in the memory management unit.

Compute Real Address

The compute real address function is used by system software to determine if a given virtual address is currently mapped in real memory, and what real address is assigned to the virtual address if it is mapped.

The compute real address function is invoked by an I/O write to the address associated with this function. Bits (0-31) of the data transferred by the I/O write instruction are used as the effective address. This effective address is then used for the normal translation process, except the results of translation are loaded into the translated real address register (TRAR), rather than being used to access memory. The TRAR contains a bit indicating whether the translation was successful, and the corresponding real memory address if the translation was successful. The indication of successful translation ignores the segment protection and present bits, as well as normal memory protection processing and lockbit processing. Results of the compute real address function are obtained by an I/O read of the TRAR.

I/O Address Assignments

A 64K-byte block of I/O addresses are assigned to the memory management unit. This 64K-byte block begins at an I/O address specified by the I/O base address register (see "I/O Base Address Register"). The I/O base address is defined to be on 64K boundaries. The I/O address assignments listed below are displacements in the specified 64K-byte block. The absolute I/O address is equal to the I/O base address plus the hex displacement as shown below.

Hex Displacement	Assignment
0000-000F	Segment Registers 0 through 15
0010	I/O Base Address Register
0011	Memory Exception Register
0012	Memory Exception Address Register
0013	Translated Real Address Register
0014	Transaction ID Register
0015	Translation Control Register
0016	RAM Specification Register
0017	ROM Specification Register
0018	RAS Mode Diagnostic Register
0019-001F	Reserved
0020-002F	TLB0 Address Tag Field for TLB0 entries 0 through 15
0030-003F	TLB1 Address Tag Field for TLB0 entries 0 through 15
0040-004F	TLB0 Real Page Number, Valid Bit, and Key Bits for TLB0 entries 0 through 15
0050-005F	TLB1 Real Page Number, Valid Bit, and Key Bits for TLB0 entries 0 through 15
0060-006F	TLB0 Write Bit, Transaction ID, and Lockbits for TLB0 entries 0 through 15
0070-007F	TLB1 Write Bit, Transaction ID, and Lockbits for TLB0 entries 0 through 15
0080	Invalidate Entire TLB
0081	Invalidate TLB Entries In Specified Segment
0082	Invalidate TLB Entry For Specified Effective Address
0083	Load Real Address
0084-0FFF	Reserved

1000-2FFF	Reference and Change bits for pages 0 through 8191
3000-FFFF	Reserved.

I/O Base Address Register Initialization

Before any control registers in the memory management unit can be accessed, the I/O base address register must be initialized to define which I/O addresses are assigned to the memory management unit. Initializing the I/O base address register gives addressability to the other control registers in the memory management unit. The address of the various control registers are defined in "I/O Address Assignments" on page 11-136.

I/O address X'808000' is reserved for initialization purposes, and provides access to the I/O base address register. System software must first execute an I/O write instruction to this address to define which I/O addresses are assigned to the memory management unit. Since the memory management unit I/O base address register is loaded with all zeros at POR, the memory management unit recognizes all valid I/O addresses in the first 64K of the I/O address space until a different value is written into the I/O base address register using an IOW instruction. Once the I/O base address register contents are defined, the other I/O registers in the memory management unit can be initialized by I/O write instructions to the individual registers. Once the contents of the I/O base address register are defined, the I/O base address register is addressable at two separate I/O addresses; at address X'808000' and at the address within the specified I/O address range of the memory management unit.

Memory Management Unit Control Register Initialization

Once the I/O base address register has been initialized, system software can then initialize other memory management unit control registers using I/O write instructions to the address of the individual control registers. System software will normally first define the actual size and beginning address of RAM and ROM. The memory management unit will initially accept any memory request, and direct it to RAM or ROM. The determination of whether the request is to RAM or ROM is based on whether the first access after POR was a read or write. If the first access was a read, all memory requests are directed to ROM. If the first access was a write, all memory requests are directed to RAM. Master mode is disabled by initializing either the RAM specification register or the ROM specification register. System software must insure that the address defined in the ROM specification register or the RAM specification register provides access to the instructions being used for initialization purposes. Once an I/O write is executed to define the ROM specification register or the RAM specification register, master mode is disabled, and only memory addresses within the range specified by the initialized specification register is be accepted. Incorrect specification of the ROM specification register or RAM specification register can result in system lock-up, since the memory management unit will no longer recognize addresses generated by the system processor. An example of the proper initialization sequence of the ROM specification register and RAM specification register is given below.

Assuming that the memory management unit is initialized to master mode, and ROM is used to contain the initialization code, the address map recognized by the memory management unit before the ROM specification register or the RAM specification register is initialized is shown in

Figure 11-36 part A. The ROM image is repeated at all addresses in the memory map. For example, with a 64K-byte ROM, the 64K-byte ROM image repeats every 64K-bytes. This means that the first word in the ROM can be accessed at address X'000000', X'010000', X'020000', . . . X'FF0000', and the second word at X'000004', X'010004', X'020004', . . . X'FF0004', with the last word at addresses X'00FFFC', X'01FFFC', X'02FFFC', . . . X'FFFFFC'. Since the ROM is 32 bits wide, the byte address of consecutive words differ by four.

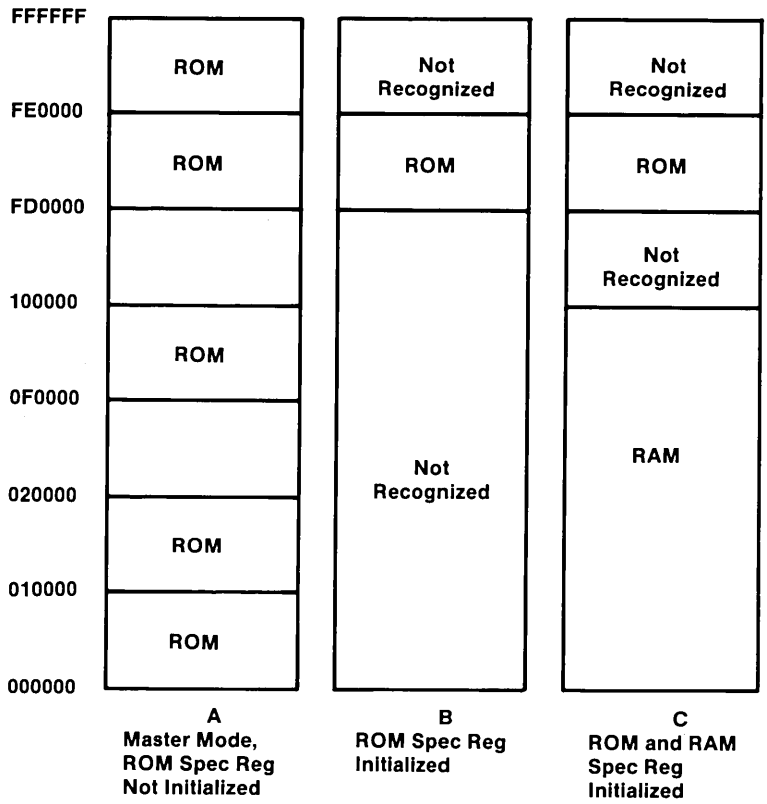


Figure 11-36. Address Space Recognized by Memory Management Unit During Initialization

The initialization code in ROM must first define the actual ROM address space by writing the appropriate size and base address information to the ROM specification register. Assuming that the ROM size is 64K- bytes, and it is desired to start the ROM address space at X'FD0000', writing the ROM specification register with this information would change the address map recognized by the memory management unit to that shown in Figure 11-36 part B. Be aware that addresses which were previously recognized by nature of master mode, are no longer recognized. The initialization

code contained in ROM must be written assuming it is at the address range specified by the value it writes to the ROM specification register. If the IAR used by the system processor for fetching subsequent instructions after the I/O write of the ROM specification register is not within the range defined by the ROM specification register, the memory management unit will fail to accept any instruction fetches and the system processor will time-out and machine check. The initial IAR contained in ROM location 000000 should contain the address of instructions within the ROM address range written to the ROM specification register by these instructions.

After initializing the ROM specification register, the RAM specification register can be initialized to define the RAM address range recognized by the memory management unit. Once the ROM specification register and RAM specification register have been initialized, the memory management unit accepts memory requests for addresses within each range, and will appropriately direct that request to either ROM or RAM. Assuming that 1M-byte of RAM is defined starting at address X'000000', the memory address map recognized by the memory management unit after initialization of the RAM specification register is shown in Figure 11-36 part C. RAM should be defined starting at address X'000000', since the system processor PSW vectors require RAM at addresses X'000100' through X'000190'.

For example, after POR, loading ROM location 0 with X'E80004' causes the system processor to begin execution with the instruction contained in the second word of the ROM. If that first instruction is a branch absolute to location X'E80008', the system processor continues execution with the instruction located in the third ROM word. The ROM starting address in the ROM specification register can then be loaded with X'E8' while still allowing the memory management unit to accept ROM instruction fetches. Once the ROM specification register is loaded, the RAM specification register can be safely loaded with the proper values for the system configuration. At that point, RAM accesses are accepted by the memory management unit.

Note: Because of the POR sequence just described, ROM must be able to recognize two address ranges, X'00XXXX' and X'E8XXXX'. This means that if address checking is performed for ROM addresses, only those bits actually used to access the ROM are checked for their proper value. The high order address bits are checked by the memory management unit ROM specification register.

ECC Checking

The memory management unit implements 32/40 ECC which detects and corrects all single bit failures, and detects all double bit failures. In addition, the 32/40 code detects several multiple bit failures not detected by a 32/39 code. ECC check bits are computed as an exclusive OR of the data bits shown in Figure 11-37.

Data				
Byte 0 Bits 0-7	**Byte 1 Bits 0-7**	**Byte 2 Bits 0-7**	**Byte 3 Bits 0-7**	**ECC Check Bit**
10101010	10101010	11000000	11000000	ECC0
01010101	01010101	00110000	00110000	ECC1
11111111	00000000	00001100	00001100	ECC2
00000000	11111111	00000011	00000011	ECC3
11000000	11000000	11111111	00000000	ECC4
00110000	00110000	00000000	11111111	ECC5
00001100	00001100	10101010	10101010	ECC6
00000011	00000011	01010101	01010101	ECC7

Figure 11-37. Check Bit Matrix For 32/40 ECC

1 = bit used in check bit generation
0 = bit not used in check bit generation

Appendix A. Instruction Mnemonics

Instruction Set By Mnemonic

The following table lists the instruction set by mnemonic and provides information for instruction execution time. These are typical execution times, and do not include potential degradation due to concurrent DMA operations.

Mnemonic	Op-Code	Format	Instruction	Cycles	Note
A	E1	(R)	Add	1	
ABS	E0	(R)	Absolute	2 (1)	7
AE	F1	(R)	Add Extended	1	
AEI	D1	(D)	Add Extended Immediate	1	
AI	C1	(D)	Add Immediate	1	
AIS	90	(R)	Add Immediate Short	1	
BALA	8A	(BA)	Branch And Link Absolute	5-7	1
BALAX	8B	(BA)	Branch and Link Absolute With Execute	5-7	1
BALI	8C	(BI)	Branch And Link Immediate	5-7	1
BALIX	8D	(BI)	Branch And Link Immediate With Execute	5-7	1
BALR	EC	(R)	Branch And Link	5-7	1
BALRX	ED	(R)	Branch And Link With Execute	5-7	1
BB	8E	(BI)	Branch On Condition Bit Immediate		
			Unsuccessful	1	
			Successful	5-7	1
BBR	EE	(R)	Branch On Condition Bit		
			Unsuccessful	1	
			Successful	5-7	1
BBRX	EF	(R)	Branch On Condition Bit With Execute		
			Unsuccessful	1	
			Successful	5-7	1

Mnemonic	Op-Code	Format	Instruction	Cycles	Note
BBX	8F	(BI)	Branch On Condition Bit Immediate With Execute		
			Unsuccessful	1	
			Successful	5-7	1
BNB	88	(BI)	Branch On Not Condition Bit Immediate With Execute		
			Unsuccessful	1	
			Successful	5-7	1
BNBR	E8	(R)	Branch On Not Condition Bit		
			Unsuccessful	1	
			Successful	5-7	1
BNBRX	E9	(R)	Branch On Not Condition Bit With Execute		
			Unsuccessful	1	
			Successful	5-7	1
BNBX	89	(BI)	Branch On Not Condition Bit Immediate With Execute		
			Unsuccessful	1	
			Successful	5-7	1
C	B4	(R)	Compare	1	
CAL	C8	(D)	Compute Address Lower Half	1	
CAL16	C2	(D)	Compute Address Lower Half 16-Bit	1	
CAS	6	(X)	Compute Address Short	1	
CAU	D8	(D)	Compute Address Upper Half	1	
CA16	F3	(R)	Compute Address 16-Bit	1	
CI	D4	(D)	Compare Immediate	1	
CIS	94	(R)	Compare Immediate Short	1	
CL	B3	(R)	Compare Logical	1	
CLI	D3	(D)	Compare Logical Immediate	1	
CLRBL	99	(R)	Clear Bit Lower Half	1	
CLRBU	98	(R)	Clear Bit Upper Half	1	
CLRSB	95	(R)	Clear SCR Bit	4	
CLZ	F5	(R)	Count Leading Zeros	1	
D	B6	(R)	Divide Step	3 (1)	7
DEC	93	(R)	Decrement	1	
EXTS	B1	(R)	Extend Sign	1	
INC	91	(R)	Increment	1	
IOR	CB	(D)	Input/Output Read	1	

Mnemonic	Op-Code	Format	Instruction	Cycles	Note
IOW	DB	(D)	Input/Output Write	2	
JB	08-0F	(JI)	Jump On Condition Bit		
			Unsuccessful	1	
			Successful	5-7	1
JNB	00-07	(JI)	Jump On Not Condition Bit		
			Unsuccessful	1	
			Successful	5-7	1
L	CD	(D)	Load	1-6	2
LC	CE	(D)	Load Character	1-6	2
LCS	4	(DS)	Load Character Short	1-6	2
LH	DA	(D)	Load Half	1-6	2
LHA	CA	(D)	Load Half Algebraic	1-6	2
LHAS	5	(DS)	Load Half Algebraic Short	1-6	2
LHS	EB	(R)	Load Half Short	1-6	2
LIS	A4	(R)	Load Immediate Short	1-6	2
LM	C9	(D)	Load Multiple	3	
LPS	D0	(D)	Load Program Status	12-16	6
LS	7	(DS)	Load Short	1-6	2
M	E6	(R)	Multiply Step	4 (1)	7
MC03	F9	(R)	Move Character Zero From Three	1	
MC13	FA	(R)	Move Character One From Three	1	
MC23	FB	(R)	Move Character Two From Three	1	
MC33	FC	(R)	Move Character Three From Three	1	
MC30	FD	(R)	Move Character Three From Zero	1	
MC31	FE	(R)	Move Character Three From One	1	
MC32	FF	(R)	Move Character Three From Two	1	
MFS	96	(R)	Move From SCR	2	
MFTB	BC	(R)	Move From Test Bit	1	
MFTBIL	9D	(R)	Move From Test Bit Immediate Lower Half	1	
MFTBIU	9C	(R)	Move From Test Bit Immediate Upper Half	1	
MTS	B5	(R)	Move To SCR	3 (4)	7
MTTB	BF	(R)	Move To Test Bit	1	
MTTBIL	9F	(R)	Move To Test Bit Immediate Lower Half	1	
MTTBIU	9E	(R)	Move To Test Bit Immediate Upper Half	1	

Mnemonic	Op-Code	Format	Instruction	Cycles	Note
N	E5	(R)	And	1	
NILO	C6	(D)	And Immediate Lower Half Extended Ones	1	
NILZ	C5	(D)	And Immediate Lower Half Extended Zeroes	1	
NIUO	D6	(D)	And Immediate Upper Half Extended Ones	1	
NIUZ	D5	(D)	And Immediate Upper Half Extended Zeroes	1	
O	E3	(R)	OR	1	
OIL	C4	(D)	OR Immediate Lower Half	1	
OIU	C3	(D)	OR Immediate Upper Half	1	
ONEC	F4	(R)	Ones Complement	1	
S	E2	(R)	Subtract	1	
SAR	B0	(R)	Shift Algebraic Right	1	
SARI	A0	(R)	Shift Algebraic Right Immediate	1	
SARI16	A1	(R)	Shift Algebraic Right Immediate Plus Sixteen	1	
SE	F2	(R)	Subtract Extended	1	
SETBL	9B	(R)	Set Bit Lower Half	1	
SETBU	9A	(R)	Set Bit Upper Half	1	
SETSB	97	(R)	Set SCR Bit	4	
SF	B2	(R)	Subtract From	1	
SFI	D2	(D)	Subtract From Immediate	1	
SIS	92	(R)	Subtract Immediate Short	1	
SL	BA	(R)	Shift Left	1	
SLI	AA	(R)	Shift Left Immediate	1	
SLI16	AB	(R)	Shift Left Immediate Plus Sixteen	1	
SLP	BB	(R)	Shift Left Paired	1	
SLPI	AE	(R)	Shift Left Paired Immediate	1	
SLPI16	AF	(R)	Shift Left Paired Immediate Plus Sixteen	1	
SR	B8	(R)	Shift Right	1	
SRI	A8	(R)	Shift Right Immediate	1	
SRI16	A9	(R)	Shift Right Immediate Plus Sixteen	1	
SRP	B9	(R)	Shift Right Paired	1	
SRPI	AC	(R)	Shift Right Paired Immediate	1	
SRPI16	AD	(R)	Shift Right Paired Immediate Plus Sixteen	1	
ST	DD	(D)	Store	2-6	4
STC	DE	(D)	Store Character	2-6	4

Mnemonic	Op-Code	Format	Instruction	Cycles	Note
STCS	1	(DS)	Store Character Short	2-6	4
STH	DC	(D)	Store Half	2-6	4
STHS	2	(DS)	Store Half Short	2-6	4
STM	D9	(D)	Store Multiple	5	
STS	3	(DS)	Store Short	2-6	
SVC	C0	(D)	Supervisor Call	16-20	6
TGTE	BD	(R)	Trap If Register Greater Than Or Equal		
			Unsuccessful	2	
			Successful	16	
TI	CC	(D)	Trap On Condition Immediate		
			Unsuccessful	2	
			Successful	16	
TLT	BE	(R)	Trap If Register Less Than		
			Unsuccessful	2	
			Successful	16	
TSH	CF	(D)	Test And Set Half	1-6	2
TWOC	E4	(R)	Twos Complement	1	
WAIT	F0	(R)	Wait	1	
X	E7	(R)	Exclusive OR	1	
XIL	C7	(D)	Exclusive OR Immediate Lower Half	1	
XIU	D7	(D)	Exclusive OR Immediate Upper Half	1	

Notes:

1. Execution cycles for successful branch instructions and jump instructions depend on system memory conditions, and the alignment of the branch or jump target instruction. Execution cycles do not include any TLB reload times if there is a TLB miss. Actual execution cycles are determined by the following conditions:

   - Execution time is 5 cycles if the target instruction is in a nonbusy memory bank, and is either a 2-byte instruction, or is a 4-byte instruction aligned on a full-word boundary.
   - Execution time is 6 cycles if the target instruction is in a busy memory bank, and is either a 2-byte instruction, or is a 4-byte instruction aligned on a full-word boundary. Execution time is also 6 cycles if the target instruction is in a nonbusy memory bank and is a 4-byte instruction not aligned on a full-word boundary.
   - Execution time is 7 cycles if the target instruction is in a busy memory bank, and is a 4-byte instruction not aligned on a full-word boundary.

2. Execution cycles for load instructions and Test and Set Half (TSH) depends on whether address translation is enabled, and which subsequent instruction uses the load (or TSH) data if address translation is disabled. Execution cycles also depend on whether the load instruction accesses data in system memory or on the I/O channel. Actual execution cycles are determined by the following conditions:

   - If address translation is enabled, using the original processor board, the system processor unconditionally waits for a response from the memory management unit before executing the next instruction. This is necessary to guarantee that the load instruction can be restarted if there is a memory management exception. In this case, the load instruction executes in 5 cycles if the load operand is in a nonbusy memory bank, and 6 cycles if the bank is busy. These times do not include any TLB reload times, if there is a TLB miss.
   - If address translation is disabled, using either processor board, the system processor continues to execute subsequent instructions while the load is being processed by the memory management unit, if the subsequent instructions do not require data from the load instruction. If the load data is required, the system processor waits until the data is returned by the memory management unit before the next instruction is executed. The load data is available after 5 cycles if the load operand is in a nonbusy bank, and 6 cycles if the bank was busy. This results in a minimum load execution time of 1 cycle if no subsequent instructions use the load data before the 5 (or 6) cycles have elapsed. For example, assume there is a load to a nonbusy memory bank, and there are three single-cycle instructions after the load which do not use the load data, and the fourth instruction does use the load data. In this case, the load executes in one cycle, and the next three instructions execute in one cycle each. However, the fourth instruction that requires the load data must wait for 2 cycles for the load to complete. In this case, the load appears to execute in 3 cycles (1 for the load plus 2 waits). Additional cycles can be required for Loads and Stores if the previous instruction is held off due to memory or PMUC contention.
   - If the load instruction references data in an adapter on the I/O channel, execution time is dependent on the speed of the adapter. For typical adapters, the load instruction executes in 9 to 11 cycles for the original processor and 15 to 20 cycles for the advanced processor.

3. Execution cycles for Load Multiple (LM) depend on whether address translation is enabled, and the number of registers loaded by the LM instruction. The number of registers loaded by the LM instruction is represented by R in the following equations.

   - The number of execution cycles is independent of translation mode.
     - 3 + 2*R When R is even.
     - 4 + 2*R When R is odd

     These times do not include any TLB reload times, if there is a TLB miss. Refer to Appendix B for serialization and synchronization effects caused by the advanced processor.

4. Execution cycles for store instructions depend on whether address translation is enabled or disabled. If address translation is disabled, execution time for store instructions is 2 cycles. If address translation is enabled, execution time for store instructions is 5 cycles if the store is to a nonbusy memory bank, and 6 cycles if it is to a busy bank. Store execution times with address translation enabled do not include any TLB reload times, if there is a TLB miss. The execution time for Stores can increase by 0, 1, or 2 cycles due to I-fetch traffic.
5. Execution cycles for Store Multiple (STM) depend on whether address translation is enabled, and the number of registers stored by the STM instruction. The number of registers stored by the STM instruction is represented by R in the following equations. Actual execution cycles are:
   - 3 + 3*R If address translation is enabled.
   - 2 + 2*R If address translation is disabled.

   These times do not include any TLB reload times, if there is a TLB miss when address translation is enabled. Refer to Appendix B for serialization and synchronization effects caused by the advanced processor.
6. Execution cycles for Load Program Status (LPS) and Supervisor Call (SVC) varies based on previous instruction. Minimum and maximum execution cycles are shown for both LPS and SVC. Refer to Appendix B for serialization and synchronization effects caused by the advanced processor.
7. Execution cycles required with the advanced processor board are listed in parenthesis.

Privileged Instructions

Mnemonic	Op-Code	Format	Instruction
CLRSB	95	(R)	CLEAR SCR BIT
LPS	D0	(D)	LOAD PROGRAM STATUS
MFS	96	(R)	MOVE FROM SCR
MTS	B5	(R)	MOVE TO SCR
SETSB	97	(R)	SET SCR BIT
WAIT	F0	(R)	WAIT

Note: Clear SCR Bit (CLRSB), Move From SCR (MFS), Move To SCR (MTS), and Set SCR Bit (SETSB) are privileged if the referenced SCR is the Counter Source (SCR 6), Counter (SCR7), Timer Status (SCR8), Machine Check Status (SCR 11), Program Check Status (SCR 11), Interrupt Request Buffer (SCR 12), Instruction Address Register (SCR 13), or the Interrupt Control Status (SCR 14). Clear SCR Bit (CLRSB), Move From SCR (MFS), Move To SCR (MTS), and Set SCR Bit (SETSB) are unprivileged if the referenced SCR is the Multiplier Quotient (SCR 10), or the Condition Status (SCR 15).

Appendix B. Advanced Processor Board

This appendix describes the changes made to the original processor board to make it an advanced processor board. Use this appendix along with Section 11 of this manual. Since only the advanced processor board changes are discussed in this appendix, you will have to refer to Section 11 for the information that remained the same as on the original processor board.

Processor Module

The following information summarizes the changes incorporated in the advanced processor module.

Overlapped Load and Store Operations

For improved system performance in a virtual memory system, the advanced processor overlaps subsequent instruction execution with memory and I/O accesses whenever possible. In the event of an exception during a memory access (such as page fault) or I/O access, the advanced processor saves information concerning the memory or I/O operations in system memory. Once system software has processed the exception, the failing memory or I/O operations can be restarted by executing an Load Program Status (LPS) instruction with restart enabled. Execution of the LPS instruction with restart enabled causes the advanced processor to restart the previously failing memory or I/O operations before subsequent instruction execution is resumed.

Memory and I/O Operation Restart

Since the advanced processor overlaps memory and I/O accesses with subsequent instruction execution whenever possible, the machine state can be altered between the time a memory or I/O operation occurs, and the time an exception is detected. The exception can be due to various reasons such as page fault, memory protection violation, or generation of an invalid memory or I/O address. Since the machine state has potentially been modified by the time the exception is detected, the failing instruction is not guaranteed to be restartable. The machine state can be modified by execution of subsequent instructions that modify the Instruction Address Register (IAR) or any registers used to compute the data or effective address for the memory or I/O operation.

When supporting a virtual memory system, it is necessary to restart failing memory operations after software has resolved an exception. The advanced processor provides hardware that saves sufficient information for each memory or I/O access to allow that access to be restarted at a later time, if the access fails. This is accomplished by saving control information defining the type of operation (memory or I/O, load or store, operation length, memory operation type) the effective address, the register number, and data for store operations in memory for each failing operation.

When any memory or I/O access instruction is executed, the advanced processor saves various information concerning the operation in internal registers. When an exception is detected (due to either an exception reply, no ACK/NAK response to a request, a nonzero high-order byte in 24-bit addressing mode, or a nonzero high-order byte in the 32-bit effective I/O address), the advanced processor determines which operations were in progress, and which need to be restarted once the exception has been resolved. Notice that this implementation of the advanced processor allows up to two memory or I/O operations to be in progress at any given time. As a result, up to two storage or I/O operations need to be restarted when an exception is detected. The advanced processor saves the proper restart information in memory when an exception is detected to guarantee that once the restart is complete that the machine state is the same as if no exceptions occurred.

For example, if a store is followed by a load to the same address, the store can fail due to a store protect violation (lockbits for processor channel DMA segments) and the load can successfully complete. This causes the load to read unmodified data from memory. The advanced processor does not load this unmodified data into the destination register. The advanced processor detects cases such as this, and restarts both the store and load after the exception is resolved to guarantee that operations appear to execute in order.

In the case of Load Multiple (LM) and Store Multiple (STM), the advanced processor holds off until all registers have been loaded or stored (subsequent instructions are not executed until the LM of STM completes). As a result, none of the registers used by the LM or STM have been altered by subsequent instructions. When an LPS with restart enabled is executed and an LM or STM is restarted, the restart uses the data contained in the general purpose registers (GPRs). The LM or STM is restarted with the first register that previously caused an exception. In the case of a STM, the exception data saved in the third word of the exception information is not used and is undefined.

Exceptions detected during the execution of an LPS to load a new IAR, ICS, and CS are exact (the IAR in the program check old program status points to the LPS instruction). The processor status is not altered, if there is an exception, during loading of the new IAR, ICS, or CS.

If an exception occurs during execution of the restart operations for an LPS, a program check interrupt occurs, and the failing operations are saved in the exception information. The IAR and ICS that were loaded during execution of the LPS is saved in the program check old program status.

Notice that not all combinations of operation length and memory operations are valid (an operation length of halfword test and set with a memory operation of store multiple is illegal). Only valid combinations of operation length and memory operation are saved in the exception information by the advanced processor. If an LPS instruction with restart enabled is executed with an invalid combination of operation length and memory operation, the restart does not occur for that operation. This does not prevent other restart operations with valid combinations of operation length and memory operation from executing.

Since the advanced processor saves the status for all failing memory and I/O access operations, software is responsible for determining if the failing operations should be restarted. For example, in a nonvirtual memory system, an invalid memory address is generally an unrecoverable error, and the program should be terminated. However, in a virtual memory system the same invalid memory address can indicate that the operating system should bring in a new page. In this case, the failing memory operations should be restarted after the new page is brought in. Since the advanced processor does not distinguish between virtual memory systems and nonvirtual memory systems, software must determine when failing memory operations should be restarted.

Under certain conditions, the advanced processor reports exceptions from cancelled memory operations. A cancelled operation occurs when a read (either a memory load or an IOR) is followed by an instruction which specifies the read register as the destination, before the read data has been loaded into the register. In this case, the data from the read is no longer needed and should not be written into the register when the data returns from the load of I/O read. The advanced processor detects this condition and marks the load or I/O read operation cancelled so that the load or I/O read will not destroy the new value computed for the register. However, if there is an exception associated with the cancelled operation, the memory controller (MMU) may have saved status information which must be cleared by system software. If a cancelled memory or I/O operation occurs, the cancelled bit (bit 31) in the first exception information word is set to 1. If an operation is cancelled, system software should clear any status information saved in the memory controller and execute an LPS instruction to return from the interrupt. An LPS instruction with restart enabled can be executed and the advanced processor will restart only noncancelled operations.

To aid in restarting failing memory operations, the advanced processor provides a Load Program Status (LPS) instruction with a restart option that causes the advanced processor to restart previously failing memory operations before subsequent instruction execution is resumed. If an LPS instruction with restart enabled is executed, the advanced processor uses exception information previously saved in system memory to restart the failing memory operations. The exception control register (SCR 9) contains the count and memory address of the exception information used to restart the failing memory operations. System software needs only to resolve the exceptions and then execute an LPS instruction with restart enabled to restart the failing operations. Once the restart is completed by the advanced processor, instruction execution is resumed at the point where the exception was detected. At this point, the machine state has been restored to the state it would have been if no exceptions occurred.

When there is more than one exception in the stack, LPS restarts the operations one at a time from the oldest to the newest. If an exception is encountered on the restart of an operation, then the restart is halted without restarting any exceptions newer than the one that regenerated. At this point, the exception stack contains the exceptions not yet restarted, and the one that regenerated.

Processor Use of Memory Mapped I/O

Advanced processor systems which implement memory mapped I/O must consider the side effects of restarting failing load and store operations. In certain cases, memory mapped I/O devices may provide an implied operation (auto increment of an internal counter) with access to a particular I/O address. Systems employing such devices must be aware of the conditions which cause the advanced processor to restart failing operations, and that a single exception can result in the restart of two operations.

In order to guarantee apparent sequential execution of instructions, the advanced processor must potentially restart two memory operations when only one has failed. Up to two operations must be restarted since this implementation of the advanced processor allows up to two memory data operations to be in progress at any given time. Notice that a valid instruction sequence is a store to a specific address followed by a load to the same address. Since the advanced processor allows up to two memory data operations to be in progress, both the store and load can be executed by the advanced processor before an exception is reported for the store. The store exception can simply be a signal to a control program to copy the current page contents and then grant write access. In this case, failure to restart both the store and the load would result in the load obtaining the old (unmodified) value from memory. The advanced processor does not compare effective addresses for stores followed by loads, it simply restarts both the store and the load, if the store fails.

Note: The Test and Set Half (TSH) instruction followed by a load does not cause the load to restart even though the TSH causes a store in memory.

This potentially results in two accesses to the same memory location if one operation fails. Systems which employ memory mapped I/O devices must insure that this has no undesirable effect on the I/O device. An example of an affected I/O device is one that contains internal sequencers and supplies sequential data by reads to the same address. Systems which employ memory mapped I/O devices with these characteristics can execute any of the serializing instructions following a store to guarantee that the store completes before subsequent instructions are executed. The serializing instructions are defined in "Serialization" on page B-5.

When a store is followed by a load, and the store faults, the advanced processor forces the load onto the stack also. However, the load data is not saved on the stack or in the destination register. There are some cases when the load cannot be restarted (memory mapped I/O). The I/O interface module captures and retains the load data in the last PMUC reply register. Software must then load the data from that register into the destination register.

Memory Protection and Address Translation

The advanced processor systems which employ memory protection or address translation functions may use part of memory to contain the protection or translation tables which are automatically accessed by hardware as required. Updates (stores) to these tables must be made in a manner that guarantees the update operation has completed before a subsequent access is made using the new data. Access to these tables is normally restricted to the control program running in privileged mode (a mode in which faults during the update operation cannot occur). However, if an update is attempted in translate or protect mode and an exception is detected during the processing of the update, the value in memory will not be changed, and any subsequent accesses will use the old data. If accesses to these tables can possibly cause an exception, software should execute a serialization instruction (see "Serialization") after the update. This guarantees the update successfully completes before subsequent accesses occur.

Serialization

To restart failing instructions in the same system processor environment they were originally executed in, it is necessary to serialize certain system operations. Serialization consists of completing all logically prior advanced processor memory and I/O data operations before the next system operation occurs. The advanced processor serializes all interrupts and the execution of the following instructions:

- Load Multiple (LM) and Store Multiple (STM).
- Move to SCR (MTS), Set SCR Bit (SETSB), and Clear SCR Bit (CLRSB).
- Supervisor Call (SVC).
- Load Program Status (LPS).
- Input/Output Write (IOW).

The following events occur during serialization:

- All logically prior advanced processor memory and I/O data operations are completed.
- The normal function associated with the serialized operation is performed. In the case of instruction execution, the instruction is executed after all logically prior advanced processor memory and I/O data operations have completed. In the case of interrupts, the PSW swap is performed after all logically prior advanced processor memory and I/O data operations have completed.
- Normal instruction execution resumes.

Exception Control Register

The exception control register is SCR 9 of the system control registers. The exception control register contains the exception count and memory address for information saved by advanced processor board for failing memory and I/O operations. This information is also used by the Load Program Status (LPS) instruction when restart is enabled to restart previously failing memory or PIO operations. The exception control register is defined as follows:

Bits 0-3 Reserved

Bits 4-7 Exception count: This 4-bit field contains the exception count. Each exception causes this count to be incremented by one, and four words of exception information to be stored at the address specified by the exception address field. The exception count is decremented by one as each failing operation is successfully restarted by executing an LPS instruction with restart enabled.

Bits 8-31 Exception Address: This 24-bit field contains the real memory address for storing the exception information. Each exception causes four words of exception information to be stored in system memory starting at the address specified by this field. The exception address is decremented by four as each exception information word is stored. When failing memory or I/O operations are restarted by executing an LPS instruction with restart enabled, the exception address is incremented by four as each word of the exception address is accessed. The exception address is decremented by 16 for each failing operation that is saved, and is incremented by 16 as each failing operation is successfully restarted. The exception address is initialized to X'000200' at power-on reset.

Exception Status Information

The advanced processor board saves four words (16 bytes) of status information for each exception. This information is saved at the memory address specified by bits 8 – 31 of the exception control register. The four words of exception status are shown in Figure B-1.

Word	Definition
Word 0	Exception Control
Word 1	Exception Address

Figure B-1 (Part 1 of 2). Exception Information Words

Word	Definition
Word 2	Exception Data
Word 3	Reserved

Figure B-1 (Part 2 of 2). Exception Information Words

The first exception information word is the exception control word. The exception control word consists of the following fields.

Bits 0-15 Reserved

Bits 16-18 Register Set Number: This 3-bit field indicates which one of the eight register sets was active when the exception occurred. Since there is only one register set in this implementation, this field is set to 0.

Bit 19 Reserved

Bits 20-23 Register Number: This 4-bit field specifies which one of the 16 general purpose registers (GPRs) contains the data that caused the exception for load and store operations. For a load, this register number specifies which GPR was being loaded. For a store, this register number specifies which GPR contained the store data. For Load Multiple (LM) and Store Multiple (SM) operations, this field specifies the first register that caused the exception.

Bits 24-25 Operation Length: This 2-bit field specifies the length of the storage or I/O operation as defined below.

00 Byte
01 Halfword
10 Fullword
11 Halfword test and set.

Bits 26-28 Storage Operation: This 3-bit field specifies the type of operation that caused an exception as defined below.

000 Load
001 Load Multiple
010 PIO Read
011 Algebraic Load
100 Store
101 Store Multiple
110 PIO Write
111 Reserved

Bits 29-30 Reserved

Bit 31 Cancelled: This bit indicates whether the operation was cancelled when the exception was detected. If this bit is set to 1, the operation is cancelled and is not restarted. If this bit is set to 0, the operation has not been cancelled and is restarted.

The second exception information word is the exception address word. The exception address word consists of the following information.

Bits 0-31 Exception Address: This word contains the 32-bit effective memory address for the failing store and I/O operations. Since the I/O address space consists of 24 bits, bits 0 through 7 of the 32-bit effective address will normally be 0 for failing I/O operations. However, in the case of an I/O addressing error due to a nonzero high-order byte in the effective address, bits 0 through 7 will contain the high-order byte of the 32-bit effective address.

The third exception information word is the exception data word. The exception data word consists of the following information.

Bits 0-31 Exception data: This word contains the 32-bit data quantity for the failing storage or I/O operation if the failing operation was a single store or I/O write. This word is undefined if the failing operation is a load, Load Multiple (LM), I/O Read (IOR), or Store Multiple (STM).

The fourth exception information word is reserved.

Program Check Handler

The processor module requires that the first instructions of the program check handler be a NOOP followed by a store of register 15 and a Branch and Link Absolute (BALA) instruction. The branch target should be the following instruction. Register 15 must then be restarted.

Tag Hold Offs

The advanced processor provides two register tags which are allocated for all read and write memory operations, and all I/O read and write operations. This permits two of these operations to be in progress at any time. If an attempt is made to execute an instruction and there are already two allocated tags, advanced processor is forced into a hold off state until a tag becomes available. If a tag is allocated for any memory read, any I/O read, or memory write operation when address translation or memory protect is enabled, the tag is freed in the cycle following the reply. If the tag is allocated for any I/O write operation or a memory write operation when address translation and memory protect are disabled, the tag is freed in the cycle following a successful transfer on the processor/MMU channel (PMUC) as indicated by the ACK/NAKD response.

For example, if the instruction sequence includes three successive load operations, the first two loads are executed with no hold offs, and the third must wait for the first load to complete. In a system with 300 nanosecond memory, there would be a hold off of two cycles before the third load instruction would execute. The two cycle hold off occurs since a tag is not available until three cycles after execution of a load instruction, and only one cycle has elapsed since execution of the first load operation.

The advanced processor serializes (holds off before execution) and synchronizes (holds off after execution) under certain conditions to guarantee that failing memory operations can be properly restarted. Serialization and synchronization results in additional hold off cycles that must be included in performance calculations. Synchronization hold offs occur after execution of Load Multiple (LM), Store Multiple (STM), and Load Program Status (LPS) instructions. Execution of subsequent instructions is delayed until all memory operations required for execution of these instructions complete.

Instruction Prefetch Buffer Loop Mode

The advanced processor includes logic to capture loops of 16 bytes or less in the Instruction Prefetch Buffer (IPB). Loop mode improves the performance of small special purpose loops such as those used for block moves, or to initialize all elements in an array. The second time through the loop, the instructions are captured in the prefetch buffer and no additional instruction fetches are required during subsequent execution of instructions in the loop. This improves performance by eliminating PMUC traffic due to instruction fetches, and the branch at the end of the loop executes in one cycle (assuming the branch is successful). Loop mode operation is handled automatically by hardware, with no special action required by software.

The advanced processor automatically enters loop mode whenever a successful Jump on Condition Bit (JB) or Jump on Not Condition (JNB) instruction is encountered with a displacement of zero (X'00') to minus seven (X'F9') halfwords. This captures backwards loops containing from one to eight instructions. Loops can contain any mix of 2- and 4-byte instructions, provided the loop ends with a JB or JBN instruction with a displacement of X'00' to X'F9'. A loop ending with a JB or JNB instruction with a displacement of X'F9' will only be captured if the loop is full word aligned. Notice that any successful branch instruction within the loop terminates loop mode.

The advanced processor assumes that the TJB/JNB at the end of the loop will be successful. In cases where the JB/JNB is successful, the JB/JNB executes in one cycle. When the loop ending condition is met (the JB/JNB is successful), an additional delay of three CPU cycles plus one memory access time will be required to fetch the next instruction.

TLB Reload Performance

The time required to reload a translation look-aside buffer (TLB) entry is a function of the inverted page table (IPT) search chain length, and whether or not the TLB entry is for a processor channel DMA segment. The TLB reload time can be computed from the sum of the following components:

- Base time
- IPT search time
- Special segment time
- Page fault time
- Correctable ECC error time.

The base time is approximately equal to 500 nanoseconds.

IPT search time is a function of the IPT search chain length. If the empty bit in the HAT pointer indicates there are no entries in the IPT, the IPT search time is 0. If the empty bit in the HAT pointer indicates there are valid entries in the IPT, the IPT search time can be computed as 300 nanoseconds times the number of entries searched in the IPT.

Time is only added when the search is actually for the processor channel DMA segment.

Page fault time is added only if a page fault condition is detected during the IPT search. The page fault time is equal to 100 nanoseconds.

Correctable ECC error time is added for each correctable ECC error, which occurs during the TLB reload. Each correctable ECC error requires 100 nanoseconds for correction.

For example, consider the calculation of the TLB reload time for special segment which is successful (that is, no page fault), and a search of two IPT entries. Also, assume no correctable ECC errors occur during the reload. This results in a total TLB reload time of 500 nsec. + (300 nsec. times 2) + 200 nsec., or 1.3 microseconds.

Memory Management Unit (MMU) Serialization

The advanced processor allows multiple load and store requests outstanding. If the first request has an exception reply (due to page fault or protection violation) but the second request is successful, then these requests are processed out of order from the software point of view. Even though the first request may get restarted after the exception condition is resolved, the first in-first out (FIFO) ordering of memory requests that software expects is violated.

The Advanced Processor Memory Management Unit (MMU) contains serialization logic which causes virtual loads and stores to memory to occur in order. The serialization logic has a locked mode and an unlocked mode. When not locked, it simply monitors MMU replies of loads and stores to system requests. An advanced processor load or store is indicated by a PMUC tag of 6 or 7, and an advanced processor coprocessor load or store is indicated by a processor/MMU (PMUC) tag of 1E or 1F. When a reply is returned to the system with an exception, the MMU serialization logic goes into locked mode.

While in locked mode, MMU will do the following:

- Force exception replies
- Cancel store operations
- Monitor for IOR or IOW of the SEAR to go unlocked.

While locked, MMU will unconditionally force exception replies to all system virtual memory requests (loads, stores, and I-fetches) and coprocessor virtual memory requests. All virtual write operations to memory are cancelled, meaning that store data does not overwrite the contents of memory while in locked mode. PIO reads and writes of MMU are not affected. MMU will leave the locked mode when a PIO read or write of SEAR occurs.

The exception conditions that will cause MMU to go into serialization locked mode are:

- Page fault
- Memory protection violation (page protection in processor channel DMA)
- Lock fault (line protection in special segments)
- Segment protection violation (processor segment versus I/O segment)
- IPT specification error
- TLB specification error.

The above exceptions cause the MMU to go into serialization lock only if it is a reply to a system virtual load or store, with the exception of TLB specification error which includes instruction fetches and coprocessor requests.

I/O Interface Module

Figure 2-6 on page 2-21 shows that the processor channel is created by a piece of logic called the I/O interface module. One I/O interface module connects the system processor and memory management unit channel (PMUC) to the processor channel. A second I/O interface module provides support for the MC68881 floating-point processor that resides on the processor board.

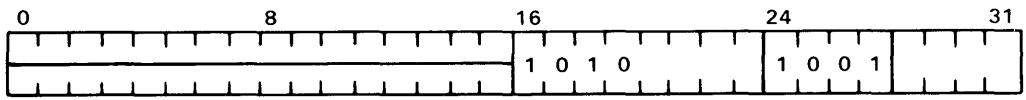
The original processor board had a block of logic analogous to the I/O interface chip 1, which generates the processor channel from the processor/MMU channel (PMUC). On the original processor board, the logic function was completely defined in the hardware. On the advanced processor board, the function is somewhat programmable. This appendix describes those programmable features. The information presentedhere applies to programming both I/O interface chips. I/O interface chip 1 creates the processor channel in the RT-RC. As such, its registers are set up by software during system initialization.

I/O interface chip 2 is programmed to support the MC68881 floating-point unit. This includes an operational mode in which the I/O interface module chip hardware assists the floating-point protocol required by the MC68881. As with I/O interface chip 1, the function is set up by system software during system initialization.

The control registers are accessed in a manner similar to the control registers on the MMU. This is via the IOR/IOW instructions to an address that matches the value in their I/O base register. The value in the I/O base register is compared with bits 8-15 of the incoming address. If a match is detected and bits 16-27 are 0, then the offset defined by bits 28-31 addresses the particular control register in the I/O interface chip.

The I/O base register must be initialized before the rest of the registers can be accessed. On I/O interface chip 1, I/O address X'800400' will access the I/O base register at initialization time. On I/O interface chip 2, the initialization address is X'800100'.

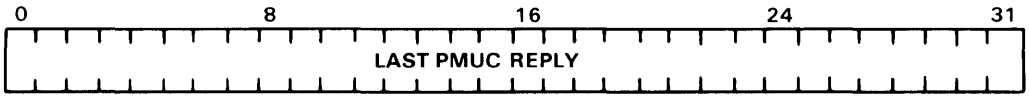
I/O Base Address Register (Offset 0)



I/O base address register bits 16-23 are compared to bits 8-15 of all incoming processor channel read and write requests. If there is a match, the request will access an I/O interface module internal control or status register. Although this is nominally an 8-bit field, the first 4 bits are fixed in hardware at B'1010'.

Bits 24-31 of the I/O base address register are compared to bits 8-15 of all incoming PMUC read and write requests. If there is a match, the request will access an I/O interface module internal control or status register. Although this is nominally an 8-bit field, the first 4 bits are fixed in hardware at B'1001'.

Last PMUC Reply Register (Offset 1)



The last PMUC reply register holds a copy of the last reply that the I/O interface module sent to the system processor. This information is necessary for certain error recovery procedures. The register contains the unformatted reply which was sent to the processor channel, so software needs to format it to form the value that would be placed in the system processor register for load halfword or load character instructions.

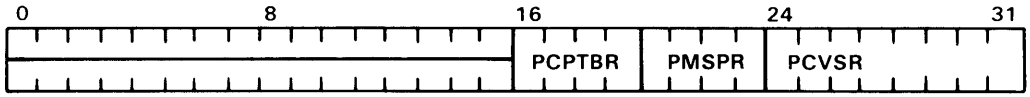
For halfword loads, if bit 30 of the load effective address is 0, then the load data is located in reply register bits 0-15. If bit 30 of the load effective address is 1, then the load data is loaded in reply register bits 16-31.

For byte loads, the value of bits 30 and 31 of the load effective address determine where the load data is located in the reply register. The value of bits 30 and 31 and the corresponding positions of the load data in the reply register is as follows:

Bits 30-31	**Load Data is in Bits**
00	0-7 (C0)
01	8-15 (C1)
10	16-23 (C2)
11	24-31 (C3)

Also, this register changes for each reply to the system processor, so the software which reads it must assure that the IOR that reads it is the first request following the error condition.

Miscellaneous Control Register (Offset 2)



Miscellaneous control register bits 16-19 are called the processor channel pass-through base register. This field is compared to bits 8-11 of incoming PMUC I/O reads and writes. If there is a match, then the request is passed through to the processor channel.

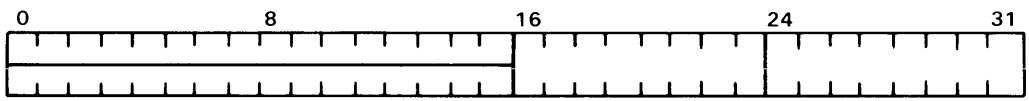
Miscellaneous control register bits 20-23 are called the PMUC segment present register. This field is compared to bits 0-3 of each incoming PMUC Load and Store requests. There must be a match for the request to be accepted.

Miscellaneous control register bits 24-31 are called the processor channel virtual segment append register. If a Load or Store request is received from the processor channel and passed to the PMUC, and the translate bit is active, then this field is sent to the PMUC as address bits 0-7. Otherwise, zeros are appended.

Subsegment Control Registers

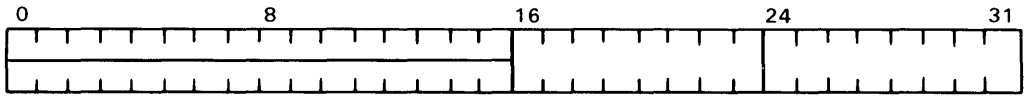
The five subsegment control registers act as a table with the five subsegment control registers sitting one on top of the other. The registers appear to be 32 bits wide, but only bits 16 through 31 are used. The registers are used by indexing or selecting a column of 5 bits, one from each register. Bits 4 through 7 of an incoming PMUC Load or Store are used to index into the table of registers. For example, if bits 4-7 equal B'0000', then bit 16 of each subsegment control register is selected as the indexed bit. These five registers control various 16-bit functions which are settable on a subsegment basis.

1. Subsegment present register (Offset 3)



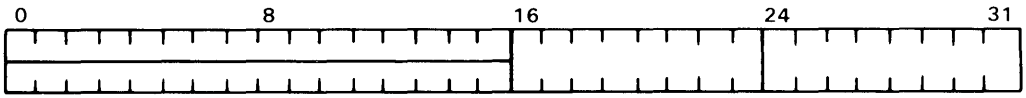
- The indexed bit must be a logical 1 for the request to be accepted.

2. Allow privileged state access (Offset 4)



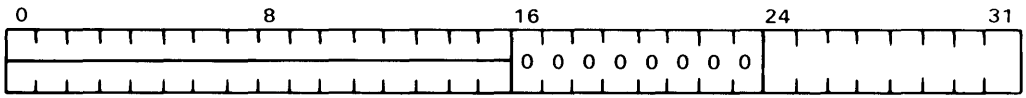
- If the incoming request came from a privileged state program, then the indexed bit must be 1 or the request will be ignored.

3. New translate bit (Offset 5)



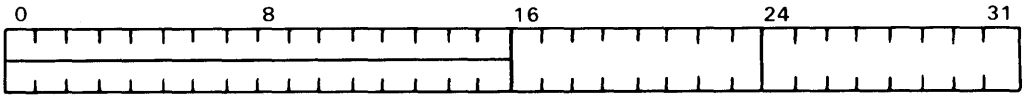
- If the incoming Load or Store is sent to the processor channel, then the indexed bit will be substituted for the translate bit which arrived on data address line 5 from the PMUC.

4. Floating point assist mode active (Offset 6)



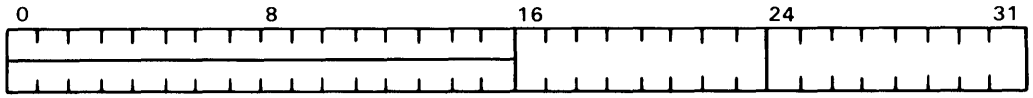
- If the indexed bit is a logical 1, then the request invokes floating-point assist mode. The first 8 bits of this register are fixed at 0. This implies that floating point assist mode cannot be activated for subsegments 0 through 7.

5. Alternate bus timing mode (Offset 7)



- If the indexed bit is a logical 1, then alternate bus timing mode is invoked. In this mode, bus timing is altered to be suitable for connecting the MC68881 and parity is not checked. If both the floating point assist mode and alternate bus timing mode bits are active for a given request, and an MC68881 FPU is connected to the I/O interface module, then hardware assist mode for an MC68881 coprocessor protocol is invoked.

General Control Register (GCR - Offset 8)

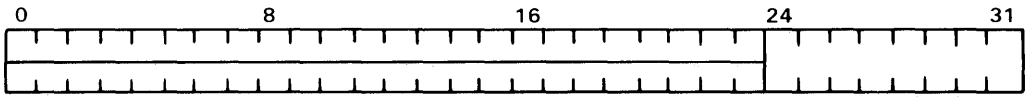


This read/write register contains miscellaneous single-bit fields for specifying I/O interface module operation, as described in the following table:

Bit	Function
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Suspends fairness in arbitration (Not supported by system)
22	Disable system board DMA read cache (See note below)
23	Reserved (I/O module 1) / + Disable processor loop mode (I/O module 2)
24	IPL ready (Read only)
25	Reserved
26	Reserved
27	Reserved
28	Disable data parity checks (This bit should be set to 1)
29	Disable serialization on error
30	2 Cycle memory (Should be set to 0 for original memory boards)
31	Enable interrupts (Not supported by system)

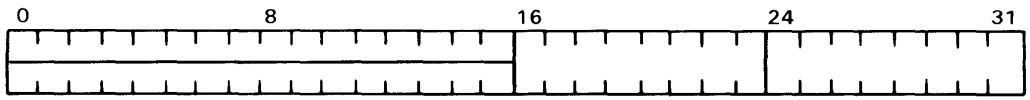
Note: Ordinarily, system board DMA reads cause double-word reads of system memory. The two words are stored in a small cache on the I/O interface module for future fast access.

Interrupt Control Register (Offset 9)



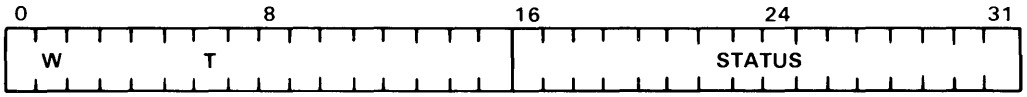
The eight active interrupt control register bits are ORed together and presented on the Interrupt Out pin of the I/O interface module, causing a level 5 processor interrupt. In a multiple processor system, this line is connected to an interrupt input of the local processor. A programmed write of this register sets the register bits if the corresponding incoming data bit is a 1. It will not reset any bits. A programmed read will read the register, then clear it.

Processor Subsegment Present Register (Offset A)



This register specifies which memory addresses appearing on the processor channel are accepted for passing to the PMUC. The 4-bit address extension from the processor channel indexes into this register, bits 16-31, and if the corresponding entry is 1, the request is accepted.

General Status Register (GSR - Offset B)



Bits 1 and 5 of the general status register apply to the address of the exception condition. Bit 1 is 1 if the request was a write. Bit 5 is 1 if the request had the translate bit active. The bits are only saved for exceptions which occurred while processing the processor channel DMA subsegment control register.

Bits 16-23 deal with requests appearing on the PMUC and sent to the processor channel. Bits 24-29 deal with requests appearing on the processor channel and sent to the PMUC, and are only preserved for incoming requests to the processor channel DMA subsegment control register.

Bits 15-31 are defined below:

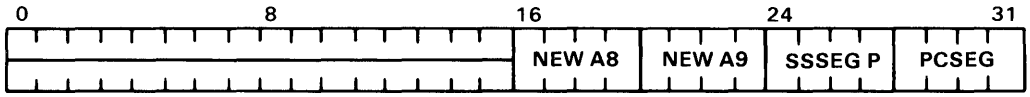
Bit	Function
15	Exception not reported for I/O Write or Real Write. (See note 2)
16	Request sent to processor channel and received no Busy signal.
17	Exception returned on a request from system processor. (See note 1)
18	Bad parity was received on PMUC data and the processor channel request was aborted.
19	Bad parity was signaled by the receiver of an outgoing processor channel request. The request was aborted.
20	Error indication detected in reply from processor channel.
21	Error detected in reply from processor channel.
22	Error signalled by system processor in attempt to return reply.
23	Reserved.
24	No response to a request sent to PMUC.

Bit	Definition
25	Exception reply to a request sent to the PMUC.
26	Parity error was signalled in response to request sent to PMUC.
27	Reserved.
28	Request disappeared on PMUC.
29	Bad parity detected on reply from PMUC.
30	Exception on request to PMUC but not reportable. (System board DMA). (See note 2)
31	Reflects state of external level 5 interrupt request.

Notes:

1. If either bit 24 or 25 is active, either a no response or an exception reply occurred for the DMA request.
2. After an exception has been returned to the system processor, no more system processor requests will be processed, attempts will receive exception replies. Normal operation resumes after the status register has been read.
3. Level 2 interrupt reported to system processor while these bits are active.

Processor Channel DMA Subsegment Control Register (Offset C)

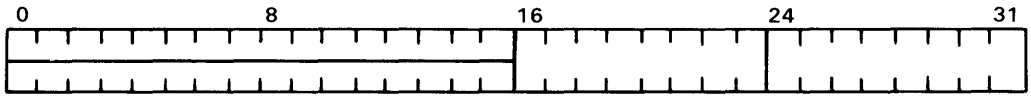


This register describes a subsegment which is used for system board DMA accesses. The subsegment facility allows system board requests to be mapped to four different processor boards at any given time. This subsegment is split into four pieces, addressed by address bits 8 and 9. The four pieces can be independently made present or absent.

Bits 0-15 of the DMA subsegment control register operations are effective only if the segment present bit for the subsegment is 0, as defined in the processor channel subsegment present register. Bits 12-15 specify which subsegment is used. Bits 8-11 are present bits for the four pieces of the subsegment. Bits 0-3 specify values for address bit 8 to be substituted for the corresponding address bit which was used to index the present bit, and bits 4-7 specify corresponding new values for address bit 9.

As mentioned, the processor channel DMA subsegment function provides support for DMA requests which come from the system board. These requests provide only a 24-bit address, so if the system board is to access multiple processor boards, extra address partitioning is necessary.

Miscellaneous Module Configuration Register (Offset D)



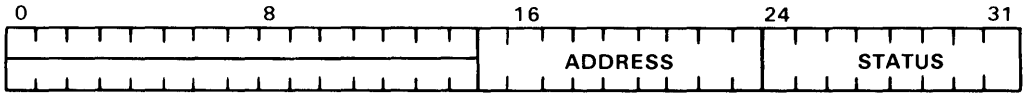
The miscellaneous module configuration register specifies the PMUC tag used on DMA operations, and the processor channel arbitration level at which the I/O interface module arbitrates. When read, it also shows the state of eight, sometimes unused, I/O interface module inputs, which are strapped to show card EC level and configuration.

Bit	Function	I/O Interface Module 1	I/O Interface Module 2
16	Module input pin sense	+ FEAT ID 5	0
17	Module input pin sense	+ FEAT ID 6	0
18	Module input pin sense	+ FEAT ID 7	0
19	Module input pin sense	− RAM on board	− MC68881 DSACK 0
20	Module input pin sense	1	− MC68881 DSACK 1
21	Module input pin sense	0	+ MC68881 ID 5
22	Module input pin sense	0	+ MC68881 ID 6
23	Module input pin sense	1	− MC68881 sense
24	Reserved		
25	PMUC Tag Select 1 (See note 2)		
26	PMUC Tag Select 2 (See note 2)		
27	Reserved		
28	Reserved		
29	Arbitration 0 (MSB)		
30	Arbitration 1		
31	Arbitration 2 (LSB)		

Notes:

1. Bits 20 through 23 represent a board ID and is strapped to a different value for each EC level of the board.
2. Bits 25 and 26 uniquely identify a particular I/O interface module for all DMA operations on the PMUC. The two I/O interface modules must be programmed to have these bits different from each other.

Floating-Point Status Register (Offset E)



Floating-point status register bits 15-21 contain bits 23-27 of any address which was sent to the PMUC on a floating-point operation and received an error. The address cannot subsequently change until the register is read. The address in the register may be different from the address that caused the error for read operations because of overlapped operation. A 1 in bit 23 indicates that the address in the register is ahead of the correct value by 4.

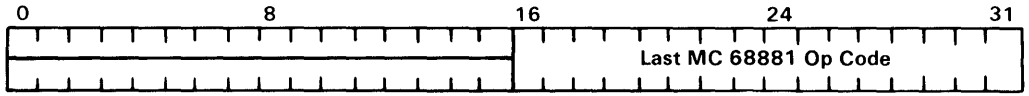
Bits 24-31 contain status information about the type of error that occurred as defined below:

Bit	Function
24	No response to a request sent to PMUC.
25	Exception reply to a request sent to the PMUC.
26	Parity error was signalled in response to request sent to PMUC.
27	Reserved.
28	Request disappeared on PMUC.
29	Bad parity detected on reply from PMUC.
30	Exception reported by MC68881.
31	Last MC68881 operation not completed.

Notes:

1. Bits 24-29 have exact counterparts in the general status register.
2. If either bit 24 or 25 is active, either a no response or an exception reply occurred for the DMA request.

Last MC68881 Op Code Register (Offset F)



This 16-bit register is loaded by the I/O interface module with the op code of the last MC68881 instruction that was started. The loading is done when the MC68881 asks for the program counter as part of the normal protocol in hardware assist operation.

The register is also written by software using an IOW to offset F.

I/O Interface Module Error Handling

This describes the error handling provided by the I/O interface module.

The I/O interface module has two status registers for preserving certain error conditions, the general status register and the floating-point status register.

Three types of transactions can be processed in which errors might be detected. These are:

- Requests which originate on the PMUC
- Requests which originate on the processor channel
- Requests which are exclusively related to floating-point assist transactions.

The first transaction type, requests which originate on the PMUC, causes the I/O interface module to become master of the processor channel and issues requests. It also includes the portions of floating point cycles which do not involve memory data transfers. If errors occur on these transactions, status is set in bits 15-23 of the general status register.

The second transaction type includes requests which originate on the processor channel and are processed by the I/O interface module. Because the I/O interface module may be acting on behalf of several originators, it is difficult to retain valid information in the event of multiple errors. For this reason, the I/O interface module assumes that most devices which can originate requests will retain their own status information, and does not ordinarily set status bits when errors occur on this type of transfer. However, it does provide status bits for one device. If an incoming subsegment is marked as special, meaning that it is probably used by the system board, status information will be retained. General status register bits 1, 5, and 24-29 contains status information about the exact nature of the problem.

The third transaction type includes the errors which occur in processing the data transfer portion of a floating-point cycle. The errors are the same as those which can occur in transmitting processor channel requests to the PMUC, so equivalent bits to general status register (24-29) are contained in the floating-point status register. Also, bits 22-29 of the address which was sent to the PMUC are saved in the floating-point status register. Because of the limited length of a floating-point data transfer, this small field is sufficient to allow the system processor to reconstruct the entire offending address in the event of a page fault.

Registers are set whenever an appropriate error occurs, and their value is retained until the register is read, at which time status bits are cleared. Figure B-2 on page B-26 and Figure B-3 on page B-28 tell the exact results of the various types of errors.

After the I/O interface module sends an exception reply to the processor, the module enters a serializing state which prevents processing of any more Load or Store requests from the processor. Instead the module returns exception replies. This serializing state is reset when the general status register (offset B) is read using an IOR instruction. The serializing state can be disabled by writing a 1 into bit 29 of the general control register (offset 8).

Symptom	Probable Cause	Action
I/O interface module wants to send exception reply to PMUC, but can't because PMUC reply not required. (Real Wrt or I/O Wrt.)		Set GSR bit 15. Activate system processor interrupt level 2.
PMUC request sent onto processor channel, no Busy signal received.	Program error.	Return reply to PMUC, with exception active. Set GSR bits 16 and 17.
PMUC request sent to processor channel, reply received from processor channel with exception active.	Program error.	Pass the exception reply through. Set GSR bit 17.
PMUC request is received. Bad parity detected on write data from PMUC.	PMUC hardware problem.	This error is detected too late to avoid starting processor channel cycle. Send request to processor channel. Activate channel error with data. Complete processor channel cycle. Accept a reply if it comes, but discard. Return exception reply to PMUC. Set GSR bits 17 and 18.
PMUC Request received, sent to processor channel. Channel error sensed.	Hardware problem.	Abort the processor channel cycle. No reply will be sent. Send exception reply to PMUC. Set GSR bits 17 and 19.
PMUC request received, sent to processor channel. Receiving device returns reply, but channel error is sensed.	Receiving device encountered hardware problem while processing request.	Send reply through to PMUC, with exception active. Set GSR bits 17 and 20.

Figure B-2 (Part 1 of 2). Errors on Requests Which Originate on PMUC

Symptom	Probable Cause	Action
PMUC request received, sent to processor channel. Receiving device returns reply, but invalid data parity sensed.	Hardware problem.	Pulse channel error. Send reply through to PMUC, with exception active. Set GSR bits 17 and 21.
PMUC request received, sent to processor channel. Reply returns OK. Error indicated in attempt to return reply to system processor.	PMUC hardware problem.	Retry reply. Set GSR bit 22.

Figure B-2 (Part 2 of 2). Errors on Request Which Originate on PMUC

Note: GSR bit 17 is only sent for exceptions to system processor requests.

Errors in Requests Originating on Processor Channel

Symptom	Probable Cause	Action
Processor channel request recognized and sent to PMUC. No response.	Software error.	Retry PMUC request twice and cancel. Return exception reply to processor channel. Set GSR bit 24.
Processor channel request accepted and sent to PMUC. Exception reply comes from PMUC.	Software error.	Pass the exception reply to processor channel. Set GSR bit 25.
Processor channel request is recognized, but channel error is sensed with data.	Hardware error.	This request is being aborted. Do not process.
Processor channel request is recognized, but parity error is detected on address or data.	Hardware error.	Activate channel error. Do not process.
Processor channel request is passed through to PMUC, parity error is signalled.	Hardware error.	Retry until timeout, send reply to processor channel with channel error. Set GSR bit 26.
Processor channel request is sent to PMUC, no reply comes back.	Hardware error.	Cancel the request. Send reply to processor channel with channel error. Set GSR bit 28.

Figure B-3 (Part 1 of 2). Errors on Request Which Originate on Processor Channel

Symptom	Probable Cause	Action
Processor channel request is sent to PMUC, invalid parity sensed on reply.	Hardware error.	Cancel the request. Send reply to processor channel with channel error. Set GSR bit 29.
Reply is sent back to processor channel, but bus master signals channel error.	Hardware error.	Ignore, bus master will handle.

Figure B-3 (Part 2 of 2). Errors on Request Which Originate on Processor Channel

Note: Status bits are only saved if the request is to the processor channel DMA control register.

I/O Interface Module Floating-Point Assist Interface

The I/O interface module implements a burst cycle for the purpose of assisting tightly coupled coprocessors, such as floating-point units.

When this cycle is engaged by a request from the PMUC, the I/O interface module performs data transfers between the floating-point unit and memory.

The floating-point assist data transfer cycle uses a special bursting DMA protocol which has been defined for the processor channel. In this context there are two operations. The floating-point load moves data from memory to the floating-point adapter using a burst write cycle. The I/O interface module is the bus master for the operation and is writing data to the floating-point adapter. The floating-point store moves data from the floating-point adapter to memory, and uses the burst read operation. The I/O interface module is the bus master and is reading data from the floating-point adapter.

Floating-point assist mode operation in the I/O interface module is invoked when a suitable request is received from the PMUC. Specifically, a request is accepted and processed as a floating-point assist mode request if the bit in subsegment control register 4 (offset 6) corresponding to the request subsegment is active. This means that floating-point assist mode can be made active for any subsegment under program control. The standard location in the RT-PC is subsegment E.

When an incoming request invokes floating-point assist mode operation, the low 24 address bits are assumed by the I/O interface module (and the AFPA) to contain control information. Several of these bits specify the data transfer. The remainder are ignored by I/O the interface module.

These control bits specify one of the following:

- The direction and length of a burst transfer which should be performed
- The transfer length register is to be written or read
- No burst transfer is to take place.

In the latter two, the PMUC request is simply passed through to the processor channel as a normal request, without a burst cycle being invoked.

If the operation is intended to create a burst transfer, then the request must be a store. The effective address for the memory access is taken from the 32-bit data field which arrives with the request.

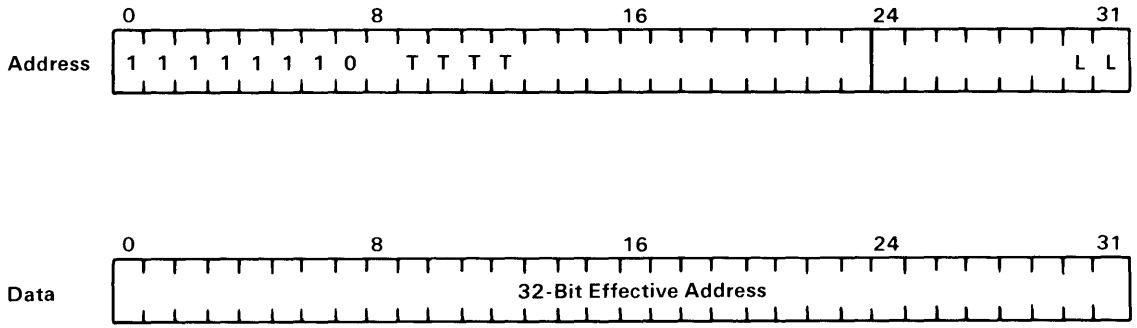


Figure B-4. PMUC Request to I/O Interface Module

Figure B-4 shows the request that is sent to the I/Ointerface module. The upper byte is decoded to determine acceptance from the PMUC and engagement of FP assist mode. Bits 4-31 are sent through to the FP adapter. In addition, the bits marked T are decoded by the I/O interface module and have the following function:

0 1 1 1	Access length register (LL must = 11)
1 x 1 1	Perform FP Store operation (move data from AFPA to memory)
0 x 1 1	Perform FP Load operation (move data from memory to AFPA, use length register)
Others	Perform FP Load operation.

The L bits normally control the transfer length and are described below:

0 0	No transfer
0 1	Transfer 1 word
1 0	Transfer 2 words
1 1	Transfer n words, where n is the content of the length register.

The I/O interface module contains a 6-bit register which specifies transfer length when LL equals 11. This register contains a binary representation of the number of words to be transferred. A register content of 0 specifies a 64-word transfer.

Floating Point Function on Advanced Processor Board

Overview of the Motorola MC68881 Floating Point Module

The MC68881 implements the full IEEE floating point standard. In addition, it provides trigonometric and transcendental functions. It allows seven data types (converting each type to extended precision for all calculations), and has eight general purpose data registers.

The MC68881 used on the Advanced Processor Board

The MC68881 requires a special communications protocol for proper operation. On the advanced processor board, this protocol is handled for some operations by the I/O interface module (hardware assist), and by software for other operations. It is important to note that even the simplest MC68881 arithmetic operation requires at least two accesses to the MC68881, writing the command register and reading the response register. For every operation, hardware assisted or not, the MC68881 is controlled via system processor initiated I/O interface module accesses to the MC68881 interface registers. All transfers of information, including floating point data, are handled by accesses to the interface registers. Direct accesses to the eight general purpose data registers are not allowed by the MC68881.

Software may access the MC68881 in two ways, either as it would a standard PIO device, referred to as software mode, or using the hardware assistance provided by the I/O interface module.

In the software mode, the program must use PIO transfers with the MC68881 and implement the MC68881 coprocessor protocol in software. The I/O interface module logic does nothing other than pass commands.

In the hardware assisted mode, a system processor Store containing the MC68881 command, specification of any operand transfer, and an address is sent to the I/O interface module. The I/O interface module manages initiation of the command and the MC68881 protocol handling with the MC68881, as well as the transfer of memory operands. System processor involvement is limited to execution of a single Store instruction.

Addressing

Access to the MC68881 is through two subsegments which may be defined by the I/O interface module program control. One of these subsegments will be set up to invoke hardware assist mode, and the other will be set up to simply pass through PIO operations. For example, the system might be set up so that hardware assisted mode would be initiated through addresses X'FCXXXXXX', and software control mode through addresses X'FDXXXXXX'.

Software Mode (PIO Mode)

In this mode, the MC68881 appears simply as a PIO device which may be written or read by the program. It appears as a set of registers which are individually addressed via bits 27-30 of the effective address. Bits 8-26 are 0. For example, if subsegment FD is used as the subsegment for software mode, the address would appear as follows:

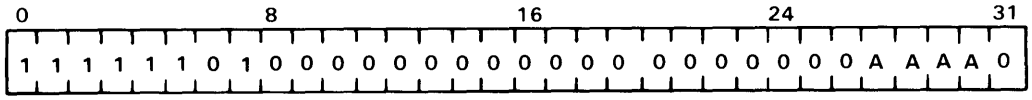


Figure B-5. PMUC Address for MC68881 - Software Mode

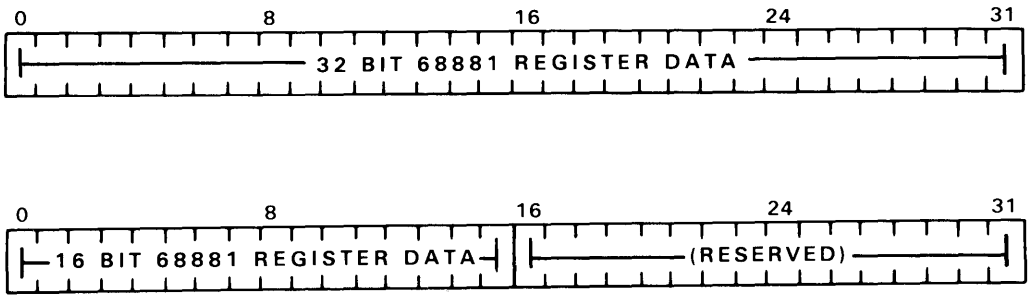


Figure B-6. PMUC Data for Read or Write of MC68881 - Software Mode

In this mode, PMUC address bits 27-30 correspond to MC68881 address bits A4-A1, and PMUC data bits 0-31 correspond to MC68881 data bits D31-D0. All figures show IBM standard bit notation (most significant bit = 0).

Please note that the MC68881 transfers all 16 bit registers on the upper 16 data bits (PMUC data bits 0-15), and therefore fullword accesses should be used on all MC68881 transfers to ensure proper operation. If halfword accesses are done, the system processor may expect (depending on the effective address used) the data to be on the lower 16 bits and therefore zeros out the upper bits containing the register contents. On writes to a 16-bit MC68881 register, the lower 16 bits should be 0; on reads, the lower 16 bits are undefined. Also note that only one register is accessed for each system processor Load or Store.

A typical protocol sequence (for instance, a memory to register add) using software mode commands to the I/O interface module would be as follows:

1. Write the command to the MC68881 command register.
2. Read the MC68881 response from the response register. Continue doing this until the MC68881 says it is ready for data.
3. Write the data to the MC68881 operand register for the required number of words. (This transfers the floating point data.)
4. Read the MC68881 response from the response register. Continue doing this until the MC68881 says it needs no further servicing.

The program may do any of the MC68881 operations using software mode. However, the overhead of implementing the MC68881 protocol is high, if done by software. Nevertheless, certain MC68881 interactions need to be performed by software control, as described later. Complete information on MC68881 protocol, individual instructions, interface and data registers is provided in the Motorola MC68881 manual.

Hardware Assist Operation

Hardware assist mode may be used for any MC68881 operation that is initiated by writing the MC68881 command register. In this mode the command is started by a system processor Store command. For example, if subsegment FC is used as the subsegment for hardware assist operations, the address and data would appear as follows:

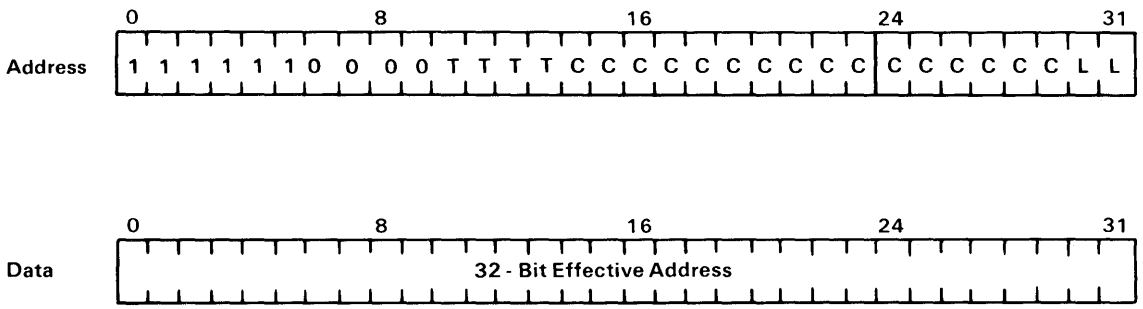


Figure B-7. PMUC Address and Data - MC68881 Hardware Assist Mode

The field marked C is the 16-bit command which will be sent to the MC68881. This command is defined in the MC68881 documentation. The field marked T is a transfer control field, and instructs the I/O interface module how to transfer data. For MC68881 operations, only three combinations are valid:

0 0 0 0 Transfer from system memory to MC68881

0 1 1 1 Transfer to or from length register (LL must = 11)

1 1 1 1 Transfer from MC68881 to system memory.

These bits work in conjunction with the length field, marked L, which defines how many words are to be transferred:

0 0 No data transfer

0 1 Transfer 1 word

1 0 Transfer 2 words

1 1 Transfer N words, N is in length register.

The length register retains whatever value is written into it until it is written again. It may also be read.

The I/O interface module manages its transfer according to this field rather than the information it receives from the MC68881 response register. In order to improve performance (and also for better commonality with the Advanced Floating-Point Accelerator) conflict between this information and the needs of the MC68881 causes an exception to be generated.

When a hardware assist operation is started by a system processor store, the I/O interface module does two things.

1. It sends the command field to the MC68881 and prefetches any operands that may be required.
2. It then carries out the MC68881 protocol, interacting with the MC68881 until no further action is required.

When the sequence is complete, a reply is sent to the system processor. If problems are encountered in the protocol, an exception reply is sent to the system processor.

A typical protocol sequence (for instance for a memory to register add) using hardware assist mode would be as follows:

1. Perform a write to the I/O interface module containing the MC68881 command and the 32-bit effective address.
2. The I/O interface module writes the MC68881 command register and begins reading from memory at the effective address.
3. The I/O interface module repeatedly reads the MC68881 response register until the MC68881 is ready for data.

4. The I/O interface module writes the MC68881 operand register using the memory data for the required number of words.
5. The I/O interface module reads the MC68881 response register until the MC68881 says it needs no further servicing.
6. The I/O interface module returns reply to the system processor if in virtual mode.

When Software Mode Should be Used

Use of hardware assist mode gives a significant performance improvement, but it cannot be used for all operations. The first class of operations for which it cannot be used are those which are not initiated by writing the MC68881 command register. This class includes Save/Restore, conditional, and control (abort) operations. Implementation of these operations is fairly complex and irregular, and could not be justified in hardware.

The second class of operations includes operations which may be very slow, such as some transcendentals. Hardware assist operation uses hardware interlocks which are not interruptable. Execution of these long operations can cause subsequent operations to be queued in hardware, with no possibility for interrupt. Locking up for an extended time period could cause system problems (interrupt latency) so it should be avoided. To solve the problem, long operations should be followed by a software mode NOP command to the MC68881, followed by repeated reads of its response register until it is no longer busy. This holds off the program until the long operation is complete without creating a hardware barrier to other processor operations. Please note that the actual long operations may be started using the hardware assist mode, but must be followed by the above softwaremode sequence.

Glossary

A. Ampere.

ac. Alternating current.

accumulator. A register in which the result of an operation is formed.

ACK0. A transmission control character for even positive acknowledgement.

ACK1. A transmission control character for odd positive acknowledgement.

active high. Designates a signal that has to go high to produce an effect. Synonymous with positive true.

active low. Designates a signal that has to go low to produce an effect. Synonymous with negative true.

adapter. An electronic part used to connect two unlike parts or machines.

address. (1) A name, label, or number identifying a location in storage, a device in a network, or any other data source. (2) A number that identifies the location of data in memory.

address bus. One or more conductors used to carry the binary-coded address from the processor throughout the rest of the system.

addressing. (1) In data communications, the way that the sending or control station selects the station to which it is sending data. (2) A means of identifying storage locations.

algorithm. A finite set of well-defined rules for the solution of a problem in a finite number of steps.

All Points Addressable (APA) display. A display that allows each pel to be individually addressed. An APA display allows for images to be displayed that are not made up of images predefined in character boxes.

alphameric. Consisting of letters, numbers and often other symbols, such as punctuation marks and mathematical symbols.

alphanumeric (A/N). Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks. Synonymous with alphameric.

alternating current (ac). A current that periodically reverses its direction of flow.

American National Standard Code for Information Exchange (ASCII). The code developed by ANSI for information interchange among data processing systems, data communications systems, and associated equipment. The ASCII character set consists of 7-bit control characters and symbolic characters.

ampere (A). The basic unit of electric current.

A/N. Alphanumeric.

analog. (1) Pertaining to data in the form of continuously variable physical quantities. (2) Contrast with digital.

array. An arrangement of elements in one or more dimensions.

asynchronous transmission. In data communications, a method of transmission in which the bits included in a character or block of characters occur during a specific time interval. However, the start of each character or block of characters can occur at any time during this interval. Contrast with *synchronous transmission.*

bandwidth. The difference, expressed in hertz, between the two limiting frequencies of a band.

base address. The beginning address for resolving symbolic references to locations in storage.

base register. A general purpose register that the programmer chooses to contain a base address.

basic assurance test (BAT). An internal diagnostic program activated each time the system is turned on.

baud. (1) A unit of signaling speed equal to the number of discrete conditions or signal events per second. For example, one baud equals one bit per second in a train of binary signals, one-half dot cycle per second in Morse code, and one 3-bit value per second in a train of signals each of which can assume one of eight different states. (2) In asynchronous transmission, the unit of modulation rate corresponding to one unit of interval per second; that is, if the duration of the unit interval is 20 milliseconds, the modulation rate is 50 baud.

binary. (1) Pertaining to a system of numbers to the base two; the binary digits are 0 and 1. (2) Involving a choice of two conditions, such as on-off or yes-no.

binary digit. (1) In binary notation, either of the characters 0 or 1. (2) Synonymous with bit.

binary notation. Any notation that uses two different characters, usually the binary digits 0 and 1.

binary synchronous communications (BSC). A form of communications line control using transmission control characters to control the transfer of data over a communications line.

bit. Either of the binary digits 0 or 1 used in computers to store information. See also *byte.*

bits per second (bps). A unit of measurement representing the number of discrete binary digits transmitted by a device in one second.

block. (1) A group of records that is recorded or processed as a unit. Same as *physical record.* (2) In data communications, a group of records that is recorded, processed, or sent as a unit. (3) A block is 1024 bytes long.

block check character. The character used in BSC to check that all bits transmitted were received.

bps. Bits per second.

branch. In a computer program an instruction that selects one of two or more alternative sets of instructions. A conditional branch occurs only when a specified condition is met.

buffer. (1) A temporary memory unit, especially one that accepts information at one rate and delivers it at another rate. (2) An area of memory, temporarily reserved for performing input or output, into which data is read, or from which data is written.

bus. One or more conductors used for transmitting signals or power.

byte. The amount of storage required to represent one character; a byte is 8 bits.

C. Celsius.

cathode ray tube (CRT). A vacuum tube in which a stream of electrons is projected onto a fluorescent screen producing a luminous spot. The location of the spot can be controlled.

CCITT. International Telegraph and Telephone Consultative Committee.

Celsius (C). A temperature scale. Contrast with Fahrenheit (F).

central processing unit (CPU). Term for processing unit.

channel. A path along which data passes. Also a device connecting the processor to I/O.

character. A letter, digit, or other symbol.

character generator. (1) In computer graphics, a functional unit that converts the coded

representation of a graphic character into the shape of the character for display. (2) In word processing, the means within equipment for generating visual characters or symbols from coded data.

character key. A keyboard key that allows the user to enter the character shown on the key. Compare with *function keys.*

character set. A group of characters used for a specific reason; for example, the set of characters a printer can print or a keyboard can support.

characters per second (cps). A standard unit of measurement for the speed at which a printer prints.

check. (1) An error condition. (2) To look for a condition.

clocking. In data communications, a method of controlling the number of data bits sent on a communications line in a given time.

CMOS. Complementary metal oxide semiconductor.

code. (1) Instructions for the computer. (2) To write instructions for the computer; to *program.* (3) A representation of a condition, such as an error code.

coding scheme. Synonym for code.

command. A request to perform an operation or execute a program. When parameters, arguments, flags, or other operands are associated with a command, the resulting character string is a single command.

communications adapter. A hardware feature enabling a computer or device to become a part of a data communications network.

complementary metal oxide semiconductor (CMOS). A logic circuit family that uses very little power. It works with a wide range of power supply voltages.

complement of a number. The value that can be added to the number to equal a given value.

concatenate. (1) To link together. (2) To join two character strings.

configuration. The group of machines, devices, and programs that make up a computer system.

conjunction. Synonym for AND operation.

connector. A part of the system unit or remote controller to which cables for display stations and printer are attached.

counter. A register or storage location used to accumulate the number of occurrences of an event.

coupler. A device connecting a modem to a telephone network.

cps. Characters per second.

CPU. Central processing unit.

CRC. Cyclic redundancy check.

CRT. Cathode ray tube.

CRT display. Cathode ray tube display.

CTS. Clear to send. Associated with modem control.

cursor. (1) A movable symbol (such as an underline) on a display, used to indicate to the operator where the next typed character will be placed or where the next action will be directed. (2) A marker that indicates the current data access location within a file.

cyclic redundancy check (CRC). (1) A redundancy check in which the check key is generated by a cyclic algorithm. (2) A system of error checking performed at both the sending and receiving station after a block-check character has been accumulated.

cylinder. All fixed disk or diskette tracks that can be read or written without moving the disk drive or diskette drive read/write mechanism.

data. (1) A representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by human or automatic means. (2) Any representations, such as characters or analog quantities, to which meaning is, or might be assigned.

data transmission. Synonym for transmission.

dc. Direct current.

Deutsche Industrie Norm (DIN). (1) German Industrial Norm. (2) The committee that sets German dimension standards.

digit. (1) A graphic character that represents an integer; for example, one of the characters 0 to 9. (2) A symbol that represents one of the non-negative integers smaller than the radix. For example, in decimal notation, a digit is one of the characters 0 to 9.

digital. (1) Pertaining to data in the form of digits. (2) Contrast with analog.

DIN. Deutsche Industrie Norm.

DIN connector. One of the connectors specified by the DIN committee.

DIP. Dual in-line package.

DIP switch. One of a set of small switches mounted in a dual in-line package.

direct current (dc). A current that always flows in one direction.

direct memory access (DMA) device. A component that can read or write to system storage directly, without processor intervention. Two device types are identified: 1) a first party DMA device resides on a hardware adapter and 2) a third party DMA device resides on the system planar. DMA capability permits simultaneous use of input/output devices and the processor.

disable. A processing unit is disabled when it prevents the occurrence of certain types of interrupts.

disabled. Pertaining to a state of a processing unit that prevents the occurrence of certain types of interruptions. Synonymous with masked.

disk. Loosely, a magnetic disk unit.

disk drive. A mechanism for moving a disk pack and controlling its movements.

diskette. A thin, flexible magnetic plate that is permanently sealed in a protective cover. It can be used to store information copies from the disk.

diskette drive. The mechanism used to read and write information on diskettes.

display. (1) A visual presentation of data. (2) A device for visual presentation of information on any temporary character imaging device. (3) To present data visually. (4) See cathode ray tube display.

display attribute. In computer graphics, a particular property that is assigned to all or part of a display; for example, low intensity, green color, blinking status.

DMA. Direct memory access.

double precision. Pertaining to the use of two computer words to represent a number in accordance with the required precision.

DSR. Data set ready. Associated with modem control.

DTR. Data terminal ready. Associated with modem control.

dual in-line package (DIP). A widely used container for an integrated circuit. DIPs have pins in two parallel rows. The pins are spaced 1/10 inch apart. See also DIP switch.

duplex. Pertains to communications data that can be sent and received at the same time. Same as *full duplex.* Contrast with *half duplex.*

duty cycle. In the operation of a device, the ratio of on time to idle time. Duty cycle is expressed as a decimal or percentage.

dynamic memory. RAM memory using transistors and capacitors as the memory elements. This memory requires a refresh (recharge) cycle every few milliseconds. Contrast with static memory.

EBCDIC. See extended binary-coded decimal interchange code.

ECC. Error checking and correction.

edge connector. A terminal block with a number of contacts attached to the edge of a printed-circuit board to facilitate plugging into a foundation circuit.

EIA. Electronic Industries Association.

enable. A processing unit is enabled when it allows certain types of interrupts.

F. Fahrenheit.

Fahrenheit (F). A temperature scale. Contrast with Celsius (C).

falling edge. Synonym for negative-going edge.

FCC. Federal Communications Commission.

feature. A programming or hardware option, usually available at an extra cost.

fetch. To locate and load a quantity of data from storage.

field. An area in a record or panel used to contain a particular catagory of data. The smallest component of a record that can be referred to by name.

fixed disk. A flat circular plate with a magnetizable surface layer on which data can be stored by magnetic recording.

fixed-disk drive. The mechanism used to read and write information on fixed disk.

flag. (1) Any of various types of indicators used for identification. (2) A character that signals the occurrence of some condition, such as the end of a word.

flexible disk. Synonym for diskette.

format. (1) A defined arrangement of such things as characters, fields, and lines, usually used for displays, printouts, or files. (2) The pattern which determines how data is recorded.

function keys. Keys that request actions but do not display or print characters. Included are the keys that normally produce a printed character, but when used with the code key produce a function instead. Compare with *character key.*

G. (1) Prefix giga; 1 000 000 000. (2) When referring to computer storage capacity, 1 073 741 824. (1 073 741 824 = 2 to the 30th power.)

gate. (1) A combinational logic circuit having one output channel and one or more input channels, such that the output channel state is completely determined by the input channel states. (2) A signal that enables the passage of other signals through a circuit.

G-byte. 1 073 741 824 bytes.

general-purpose register (GPR). An explicitly addressable register that can be used for a variety of purposes (for example, as an accumulator or an index register. accumulator, as an index register, or as a special handler of data.

giga (G). Prefix 1 000 000 000.

gram (g). A unit of weight (equivalent to 0.035 ounces).

graphics. A type of data created from fundamental drawing units such as lines, splines, curves, polygons, and so forth.

graphic character. A character that can be displayed or printed.

half-duplex. Pertains to communications in which data can be sent in only one direction at a time. Contrast with *duplex.*

hardware. The equipment, as opposed to the programming, of a computer system.

HAT. Hash anchor table.

head. A device that reads, writes, or erases data on a storage medium; for example, a small electromagnet used to read, write, or erase data on a magnetic disk.

hertz (Hz). A unit of frequency equal to one cycle per second.

hex. Common abbreviation for hexadecimal.

hexadecimal. Pertaining to a system of numbers to the base sixteen; hexadecimal digits range from 0 (zero) through 9 (nine) and A (ten) through F (fifteen).

high-order position. Most significant; leftmost. For example, bit 0 in a register.

Hz. Hertz.

immediate data. Data appearing in an instruction itself (as opposed to the symbolic name of the byte of data). The data is immediately available from the instruction and therefore does not have to be read from memory.

immediate instruction. An instruction that contains within itself an operand for the operation specified, rather than an address of the operand.

index. (1) A table containing the key value and location of each record in an indexed file. (2) A computer storage position or register, whose contents identify a particular element in a set of elements.

index register. A register whose contents may be used to modify an operand address during the execution of computer instructions.

indicator. An internal switch that communicates a condition between parts of a program or procedure.

infinity. A name for the upper boundary of the set of numbers.

inhibited. (1) Pertaining to a state of a processing unit in which certain types of interruptions are not allowed to occur. (2) Pertaining to the state in which a transmission control unit or an audio response unit cannot accept incoming calls on a line.

initialize. To set counters, switches, addresses, or contents of storage to 0 or other starting values at the beginning of, or at prescribed points in, the operation of a computer routine.

input/output (I/O). Pertaining to either input, output, or both between a computer and a device.

input-output channel controller (IOCC). A hardware component that supervises communication between the input/output channel and the processor.

instruction. A statement that specifies an operation to be performed by the computer, along with the values or locations of operands, if any exist. This statement represents the programmer's request to the processor to perform a specific operation.

instruction set. The set of instructions of a computer, of a programming language, or of the programming languages in a programming system.

interface. A shared boundary between two or more entities. An interface might be a hardware component to link two devices together or it might be a portion of memory or registers accessed by two or more computer programs.

interrupt. (1) To temporarily stop a process. (2) In data communications, to take an action at a receiving station that causes the sending station to end a transmission. (3) A signal sent by an I/O device to the processor when an error has occurred or when assistance is needed to complete I/O. An interrupt usually suspends execution of the currently executing program.

I/O. See input/output.

I/O area. Synonym for buffer.

IOCC. I/O Channel Controller. A hardware component that supervises communication

between the input/output channel and the system processor.

IPT. Inverted page table.

irrecoverable error. An error that makes recovery impossible without the use of recovery techniques external to the computer program or run.

k. Prefix kilo; 1000.

K. When referring to memory capacity, 1024. (1024 = 2 to the 10th power.)

K byte. 1024 bytes.

kg. Kilogram; 1000 grams.

kHz. Kilohertz; 1000 hertz.

kilo (k). Prefix 1000

kilogram (kg). 1000 grams.

kilohertz (kHz). 1000 hertz

latch. (1) A simple logic-circuit memory element. (2) A feedback loop in sequential digital circuits used to maintain a state.

least-significant digit. The rightmost digit. See also low-order position.

LED. Light-emitting diode.

light-emitting diode (LED). A semiconductor device that gives off visible or infrared light when activated.

load. (1) To move data or programs into memory. (2) To place a diskette into a diskette drive, or a magazine into a diskette magazine drive. (3) To insert paper into a printer.

logarithm. A mathematical operation related to the base of a numbering system.

low-order position. The rightmost position in a string of characters. See also least-significant digit.

m. (1) Prefix milli; 0.001. (2) Meter.

M. (1) Prefix mega; 1 000 000. (2) When referring to computer memory capacity, 1 048 576. (1 048 576 = 2 to the 20th power.)

mA. Milliampere; 0.001 ampere.

machine language. A language that can be used directly by a computer without intermediate processing.

mark. A symbol or symbols that indicate the beginning or the end of a field, of a word, of an item of data, or of a set of data such as a file, a record, or a block.

mask. A pattern of characters that controls the keeping, deleting, or testing of portions of another pattern of characters.

masked. Synonym for disabled.

matrix. An array arranged in rows and columns.

M byte. 1 048 576 bytes.

mega (M). Prefix 1 000 000.

megabyte. One million bytes.

megahertz (MHz). 1 000 000 hertz.

memory. Storage on electronic memory such as random access memory, read only memory, or registers.

Memory Management Unit (MMU). Hardware on the system board that manages virtual memory by providing translation from a virtual address to a real address.

meter (m). A unit of length (equivalent to 39.37 inches).

MFM. Modified frequency modulation.

MHz. Megahertz; 1 000 000 hertz.

micro (μ) Prefix 0.000 001.

microcode. (1) One or more microinstructions. (2) A code, representing the instructions of an instruction set, implemented in a part of memory that is not program-addressable.

microinstruction. (1) An instruction of microcode. (2) A basic or elementary machine instruction.

microprocessor. An integrated circuit that accepts coded instructions for execution; the instructions may be entered, integrated, or stored internally.

milli (m). Prefix 0.001.

milliampere (mA). 0.001 ampere.

millisecond (ms). 0.001 second.

MMU. See memory management unit

mnemonic. The field of an assembler instruction that contains the acronym or abbreviation for a machine instruction. Using mnemonics frees the programmer from having to remember the machine's numeric operator codes.

mode. (1) A method of operation; for example, the binary mode, the interpretive mode, the alphanumeric mode. (2) The most frequent value in the statistical sense.

modem (modulator-demodulator). A device that converts data from the computer to a signal that can be transmitted on a communications line, and converts the signal received to data for the computer.

modified frequency modulation (MFM). The process of varying the amplitude and frequency of the 'write' signal. MFM pertains to the number of bytes of storage that can be stored on the recording media. The number of bytes is twice the number contained in the same unit area of recording media at single density.

modulation. Changing the frequency or size of one signal by using the frequency or size of another signal.

modulation rate. The reciprocal of the measure of the shortest nominal time interval between successive significant instants of the modulated signal. If this measure is expressed in seconds, the modulation rate is expressed in baud.

monitor. (1) A device that observes and verifies the operation of a data processing system and indicates any significant departure from the norm. (2) Software or hardware that observes, supervises, controls, or verifies the operations of a system.

most-significant digit. The leftmost (non-zero) digit. See also high-order position.

ms. Millisecond; 0.001 second.

multiplexer. A device capable of interleaving the events of two or more activities, or capable of distributing the events of an interleaved sequence to the respective activities.

multipoint. In data communications, pertains to a network that allows two or more stations to communicate with a single system on one line.

multiprogramming. The processing of two or more programs at the same time.

n. Prefix nano; 0.000 000 001.

nano (n). Prefix 0.000 000 001.

nanosecond (ns). 0.000 000 001 second.

negative true. Synonym for active low.

negative-going edge. The edge of a pulse or signal changing in a negative direction. Synonymous with falling edge.

non-return-to-zero (inverted) recording (NRZI). Deprecated term for non-return-to-zero change-on-ones recording.

nonvolatile random access memory. A portion of random access memory that retains its contents after electrical power to the machine is shut off.

NRZI. Non-return-to-zero change-on-ones recording.

ns. Nanosecond; 0.000 000 001 second.

NUL. The null character.

null character (NUL). The character hex 00, used to represent the absence of a printed or displayed character.

NVRAM. See *nonvolatile random access memory.*

odd-even check. Synonym for parity check.

offline. Pertaining to the operation of a functional unit without the continual control of a computer.

one-shot. A circuit that delivers one output pulse of desired duration for each input (trigger) pulse.

op code. See *operation code.*

operand. An instruction field that represents data (or the location of data) to be manipulated or operated upon. Not all instructions require an operand field.

operating system. Software that controls the running of programs; in addition, an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

operation code. A numeric code indicating to the processor which operation should be performed.

OR. A logic operator having the property that if P is a statement, Q is a statement, R is a statement, . . ., then the OR of P, Q, R, . . . is true if at least one statement is true, false if all statements are false.

OR gate. A gate in which the output is 1 only if at least one input is 1.

output. The result of processing data.

output process. (1) The process that consists of the delivery of data from a data processing system, or from any part of it. (2) The return of information from a data processing system to an end user, including the translation of data from a machine language to a language that the end user can understand.

overcurrent. A current of higher than specified strength.

overflow indicator. (1) An indicator that signifies when the last line on a page has been printed or passed. (2) An indicator that is set on if the result of an arithmetic operation exceeds the capacity of the accumulator.

overrun. Loss of data because a receiving device is unable to accept data at the rate it is transmitted.

overvoltage. A voltage of higher than specified value.

page. A block of instructions, data, or both.

page fault. A program interruption that occurs when a page of memory not in real memory is referred to by an active page.

parallel. (1) Pertaining to the concurrent or simultaneous operation of two or more devices, or to the concurrent performance of two or more activities. (2) Pertaining to the concurrent or simultaneous occurrence of two or more related activities in multiple devices or channels. (3) Pertaining to the simultaneity of two or more processes. (4) Pertaining to the simultaneous processing of the individual parts of a whole, such as the bits of a character and the characters of a word, using separate facilities for the various parts. (5) Contrast with serial.

parameter. Information that the user supplies to a panel, command, or function.

picture element (pel). In computer graphics, the smallest element of a display space that can be independently assigned color and intensity.

phototransistor. A transistor whose switching action is controlled by light shining on it.

picture element (PEL). The smallest displayable unit on a display.

PMUC. processor/memory management unit

polling. A method for determining whether any of the stations sharing a communications line has data to send.

port. An access point for data input to or data output from a computer system.

positive true. Synonym for active high.

positive-going edge. The edge of a pulse or signal changing in a positive direction. Synonymous with rising edge.

power supply. A device that produces the power needed to operate electronic equipment.

precision. A measure of the ability to distinquish between nearly equal values. See *single precision* and *double precision.*

printed-circuit board. A usually copper-clad plastic board used to make a printed circuit.

priority. The relative ranking of items. For example, a job with high priority in the job queue will be run before one with medium or low priority.

privileged instructions. System control instructions that can only run in the processor's privileged state. Privileged instructions generally manipulate virtual machines or the memory manager; they typically are not used by application programmers. See *privileged state.*

privileged state. A hardware protection state in which the processor can run privileged instructions. The processor's privileged state supports the virtual machine's VRM state.

processing program. A program that performs such functions as compiling, assembling, or translating for a particular programming language.

processing unit. A functional unit that consists of one or more processors and all or part of internal memory.

processor. (1) In a computer, a functional unit that interprets and executes instructions. (2) A functional unit, a part of another unit such as a terminal or a processing unit, that interprets and executes instructions. (3) Deprecated term for processing program. (4) See microprocessor.

processor channel. The interface used directly by the system processor and the memory management unit.

program. A file containing a set of instructions conforming to a particular programming language syntax.

propagation time. The time necessary for a signal to travel from one point on a communications line to another.

protocol. (1) A specification for the format and relative timing of information exchanged between communicating parties. (2) The set of rules governing the operation of functional units of a communication system that must be followed if communication is to be achieved.

pulse. A variation in the value of a quantity, short in relation to the time schedule of interest, the final value being the same as the initial value.

RAM. Random access memory. Read/write memory.

random access memory (RAM). Read/write memory.

raster array. In computer graphics, a predetermined pattern of lines that provides uniform coverage of a display space.

read. To acquire or interpret data from a storage device, from a data medium, or from another source.

read-only memory (ROM). A memory device whose contents cannot be modified. The memory is retained when power is removed.

read/write memory. A memory device whose contents can be modified. Also called RAM.

Real Address. A 24 bit address on the processor channel which will be applied to memory by the memory management unit without modification by the translation mechanism of MMU.

real number. A number, containing a decimal point, stored in fixed-point or floating-point format.

Real Resolved Address. A real processor channel address in which the 8 high order bits as received from a DMA adapter have been replaced by the IOCC.

recoverable error. An error condition that allows continued execution of a program.

red-green-blue-intensity (RGBI). The description of a direct-drive color monitor that accepts input signals of red, green, blue, and intensity.

redundancy check. A check that depends on extra characters attached to data for the detection of errors. See cyclic redundancy check.

Region. The term region is used within the IOCC document to reference to a 128K byte block of memory on a 128 KB boundary. The MMAT and TCW table map each entry onto a region for purpose of control.

register. (1) A memory area, in a computer, capable of storing a specified amount of data such as a bit or an address. (2) See *general purpose register* and *system control register.*

register pair. See *register twin.*

register twin. The twin of a general purpose register is the binary value of the register with the low-order bit inverted. For example, the twin of register 5 with a binary value of 0101 is register 4 with a binary value of 0100.

retry. To resend the current block of data (from the last EOB or ETB) a prescribed number of times, or until it is entered correctly or accepted.

reverse video. A form of highlighting a character, field, or cursor by reversing the the color of the character, field, or cursor with its background; for example, changing a red character on a black background to a black character on a red background.

RGBI. Red-green-blue-intensity.

rising edge. Synonym for positive-going edge.

ROM. Read-only memory.

ROM/BIOS. The ROM resident basic input/output system, which provides the level control of the major I/O devices in the computer system.

RS232C. A standard by the EIA for communication between computers and external equipment.

RTS. Request to send. Associated with modem control.

schematic. The representation, usually in a drawing or diagram form, of a logical or physical structure.

second level interrupt handler (SLIH). A routine that handles the processing of an interrupt from a specific adapter. An SLIH is called by the first level interrupt handler associated with that interrupt level.

sector. (1) An area on a disk track or a diskette track reserved to record information. (2) The smallest amount of information that can be written to or read from a disk or diskette during a single read or write operation.

signal. A variation of a physical quantity, used to convey data.

sink. A device or circuit into which current drains.

Soft Reset. A system processor reset issued by the locator adapter upon detection of a unique 3-keystroke sequence.

software. (1) Computer programs, procedures, and rules concerned with the operation of a data processing system. (2) Contrast with hardware.

source. The origin of a signal or electrical energy.

SS. Start-stop.

start bit. (1) A signal to a receiving mechanism to get ready to receive data or perform a function. (2) In a start-stop system, a signal preceding a character or block that prepares the receiving device for the reception of the code elements.

start-stop system. A data transmission system in which each character is preceded by a start bit and is followed by a stop bit.

start-stop (SS) transmission. (1) Asynchronous transmission such that a group of signals representing a character is preceded by a start bit and followed by a stop bit. (2) Asynchronous transmission in which a group of bits is preceded by a start bit that prepares the receiving mechanism for the reception and registration of a character and is followed by at least one stop bit that enables the receiving mechanism to come to an idle condition pending the reception of the next character.

static memory. RAM memory using flip-flops as the memory elements. Data is retained as long as power is applied to the flip-flops. Contrast with dynamic memory.

stop bit. (1) A signal to a receiving mechanism to wait for the next signal. (2) In a start-stop system, a signal following a character or block that prepares the receiving device for the reception of a subsequent character or block.

memory. (1) A memory device. (2) A device, or part of a device, that can retain data. (3) The retention of data in a memory device. (4) The placement of data into a memory device.

STX. Start-of-text.

supervisor call (SVC). An instruction that interrupts the program being executed and passes control to the supervisor so it can perform a specific service indicated by the instruction.

symbol. (1) A conventional representation of a concept or a representation of some thing by reason of relationship, association, or convention. (2) A representation of something by reason of relationship, association, or convention.

synchronization. The process of adjusting the corresponding significant instants of two signals to obtain the desired phase relationship between these instants.

synchronous transmission. (1) Data transmission in which the time of occurrence of each signal representing a bit is related to a fixed time frame. (2) Data transmission in which the sending and receiving devices are operating continuously at substantially the same frequency and are maintained, by means of correction, in a desired phase relationship.

syntax. The rules for the construction of a command or program.

system memory. Program-addressable memory from which instructions and other data can be loaded directly into registers for subsequent execution or processing.

system unit. The part of the system that contains the processing unit, the disk drive and the disk, and the diskette drive and diskettes.

TLB. Translation look-aside buffer.

TCW. Translation Control Word used within the IOCC to translate and/or reformat the incoming DMA address.

time-out. (1) A parameter related to an enforced event designed to occur at the conclusion of a predetermined elapsed time. A time-out condition can be cancelled by the receipt of an appropriate time-out cancellation signal. (2) A time interval allotted for certain operations to occur; for example, response to polling or addressing before system operation is interrupted and must be restarted.

track. A circular path on the surface of a disk or diskette on which information is magnetically recorded and from which recorded information is read.

transistor-transistor logic (TTL). A popular logic circuit family that uses multiple-emitter transistors.

translate. To transform data from one language to another.

transmission. (1) The sending of data from one place for reception elsewhere. (2) In ASCII and data communication, a series of characters

including headings and text. (3) The dispatching of a signal, message, or other form of intelligence by wire, radio, telephone, or other means. (4) One or more blocks or messages. For BSC and start-stop devices, a transmission is terminated by an EOT character. (5) Synonymous with data transmission.

trap. An unprogrammed, hardware-initiated jump to a specific address. Occurs as a result of an error or certain other conditions.

TTL. Transistor-transistor logic.

two's complement. Representation of negative binary numbers. Formed by subtracting each digit of the number from zero, then adding one to the result.

typamatic key. A key that repeats its function multiple times when held down.

unprivileged state. A hardware protection state in which the processor can only run unprivileged instructions. The processor's unprivileged state supports the virtual machine's operating system state and problem state.

V. Volt.

video. Computer data or graphics displayed on a cathode ray tube, monitor, or display.

Virtual Address. A 32 bit address on the internal bus intended to be translated by Memory Management.

volt. The basic practical unit of electric pressure. The potential that causes electrons to flow through a circuit.

W. Watt.

watt. The practical unit of electric power.

word. A contiguous series of 32 bits (four bytes) in storage,

write. To make a permanent or transient recording of data in a storage device or on a data medium.

write precompensation. The varying of the timing of the head current from the outer tracks to the inner tracks of the diskette to keep a constant 'write' signal.

Index

A

B

C

D

E

F

G

I

K

L

M

O

P

R

S

T

U

International Business
Machines Corporation
Department 997, Building 998
11400 Burnet Rd.
Austin, Texas 78758

Printed in the
United States of America

75X0232