# IOS™ Model E System Programmer Reference Manual

CSM-1010-000

Cray Research Private

Cray Research, Inc.

Any shipment to a country outside of the United States requires a letter of assurance from Cray Research, Inc.

Requests for copies of Cray Research, Inc. publications should be directed to:

CRAY RESEARCH, INC.
Logistics
6251 South Prairie View Road
Chippewa Falls, WI 54729

Comments about this publication should be directed to:

CRAY RESEARCH, INC.
Hardware Publications and Training
770 Industrial Blvd.
Chippewa Falls, WI 54729

# Record of Revision

Each time this manual is revised and reprinted, all changes issued against the previous version are incorporated into the new version, and the new version is assigned an alphabetic level which is indicated in the publication number on each page of the manual.

Changes to part of a page are indicated by a change bar in the margin directly opposite the change. A change bar in the footer indicates that most, if not all, of the page is new. If the manual is rewritten, the revision level changes but the manual does not contain change bars.

| REVISION | DESCRIPTION |
|---|---|
| | May 1991. Original printing. |
| 001 | October 1991. This change packet adds PINT information to Section 4. |

# PREFACE

This manual describes the architecture and functions of the Cray I/O subsystem model E (IOS-E), manufactured by Cray Research, Inc. (CRI). The manual is written to help programmers and maintenance personnel write and debug programs.

The scope of this manual includes information on the basic purpose and architecture of the IOS-E. Detailed information on the I/O processors, I/O buffers, I/O channels, channel adapters, cluster and workstation interfaces, and the I/O instruction set is provided.

This manual uses the following notational conventions:

- Register and memory bit positions are numbered from right to left as powers of 2, starting with bit $2^0$.
- All numbers, except channel numbers, are decimal if they have no subscript (for example, 12).
- Channel numbers are octal (for example, channel 34).
- Additional notational conventions are used in the I/O processor instructions described in Section 7 of this manual. These conventions are described at the beginning of Section 7.
- Octal numbers (except channel numbers) are indicated with a subscript 8 (for example, $14_8$).

# CONTENTS

**5. CHANNEL ADAPTERS (continued)**

**5. CHANNEL ADAPTERS (continued)**

5. CHANNEL ADAPTERS (continued)

6. CLUSTER INTERFACE/WORKSTATION INTERFACE (continued)

FIGURES

**FIGURES (continued)**

**FIGURES (continued)**

**TABLES**

**TABLES (continued)**

# CONTENTS

**FIGURES**

**TABLES**

# 1. I/O SUBSYSTEM OVERVIEW

The Cray Research, Inc. I/O subsystem model E, hereafter referred to as the IOS, is a high-performance computer. The IOS enables a Cray Research mainframe and a Cray Research SSD solid-state storage device (SSD) to communicate with peripheral devices such as disk drives and front-end computers. The IOS performs the following functions:

- It controls all data transfers between the mainframe or SSD and peripheral devices.

- It buffers all data transfers between the mainframe or SSD and peripheral devices.

- It converts data to and from the formats used by peripheral devices.

- It detects and corrects certain types of data errors that occur during transfers.

The IOS is designed to transfer data at extremely high rates of speed while maintaining data integrity. A fully-configured IOS has a total data transfer bandwidth of over 6,800 Mbytes/s. Single-error correction/double-error detection (SECDED) logic [†] on all data paths in the IOS minimizes adverse affects of data errors.

Figure 1-1 shows the architecture of the IOS. The IOS consists of a variable number of I/O clusters and two workstation interfaces (WINs). The clusters allow the mainframe and SSD to communicate with peripheral devices. The WINs allow the operator workstation (OWS) and maintenance workstation (MWS) to control and monitor the operation of the clusters.

The remainder of this section provides a general description of the I/O clusters and WINs. A chart at the end of the section lists specifications of the major components. The other manual sections describe in detail the architecture and operation of the cluster components and WINs.

## I/O CLUSTERS

Each I/O cluster is an independent unit that enables the mainframe and SSD to communicate with a set of peripheral devices. The number of clusters in the IOS can vary between one and eight, depending on the number of peripheral devices to be connected. (The number of clusters is limited by the number of I/O channels in the mainframe.)

[†] Hamming, R. W. "Error Detection and Correcting Codes." *Bell System Technical Journal.* 29.2 (1950): 147-160.

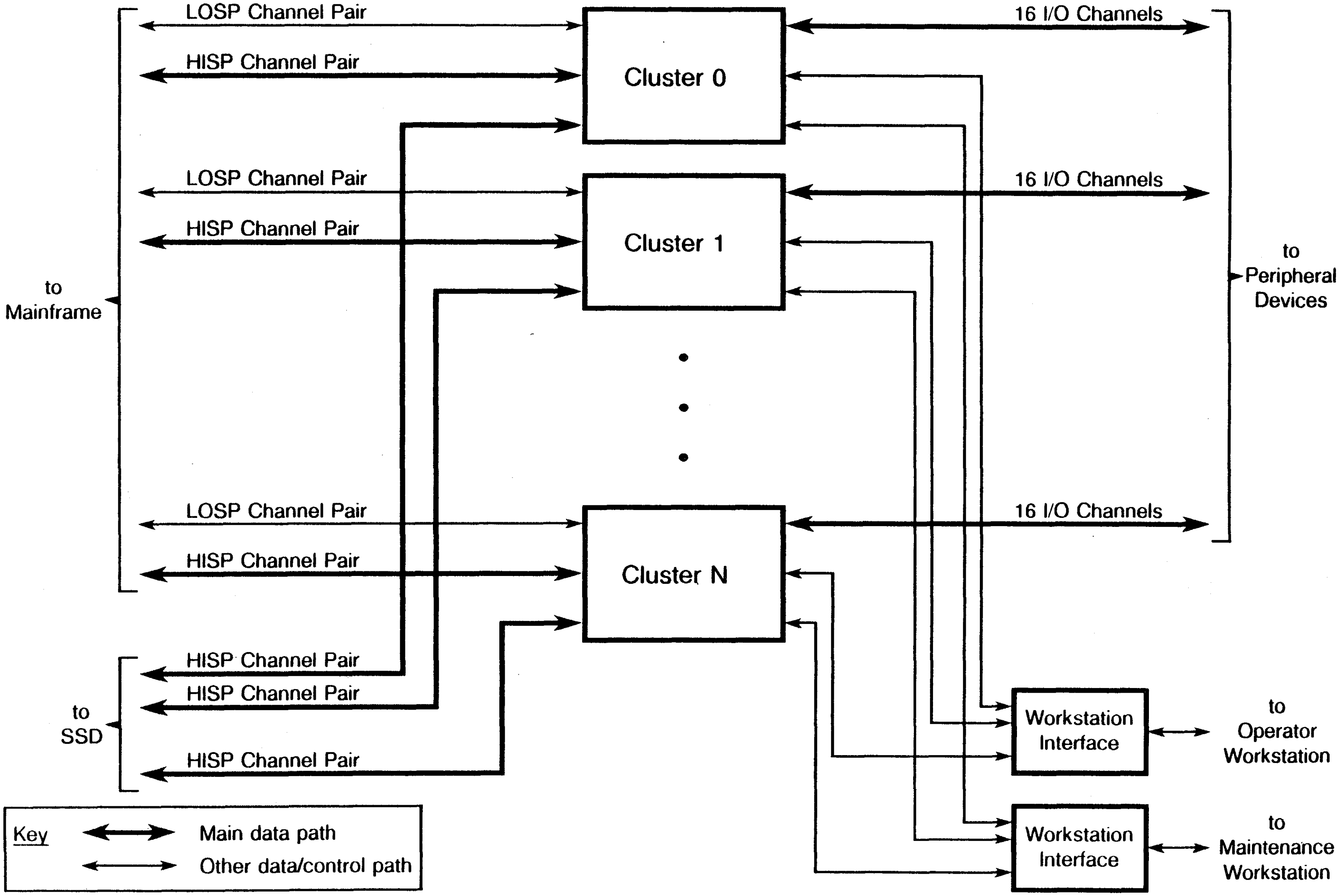


Figure 1-1. I/O Subsystem Block Diagram

Figure 1-2 shows the architecture of an I/O cluster. Each cluster contains the following components:

- One multiplexer I/O processor (IOP MUX)
- Four auxiliary I/O processors (EIOPs)
- Sixteen I/O buffers
- One low-speed (LOSP) channel pair
- Two high-speed (HISP) channel pairs
- Sixteen channel adapters
- One cluster interface (CIN)

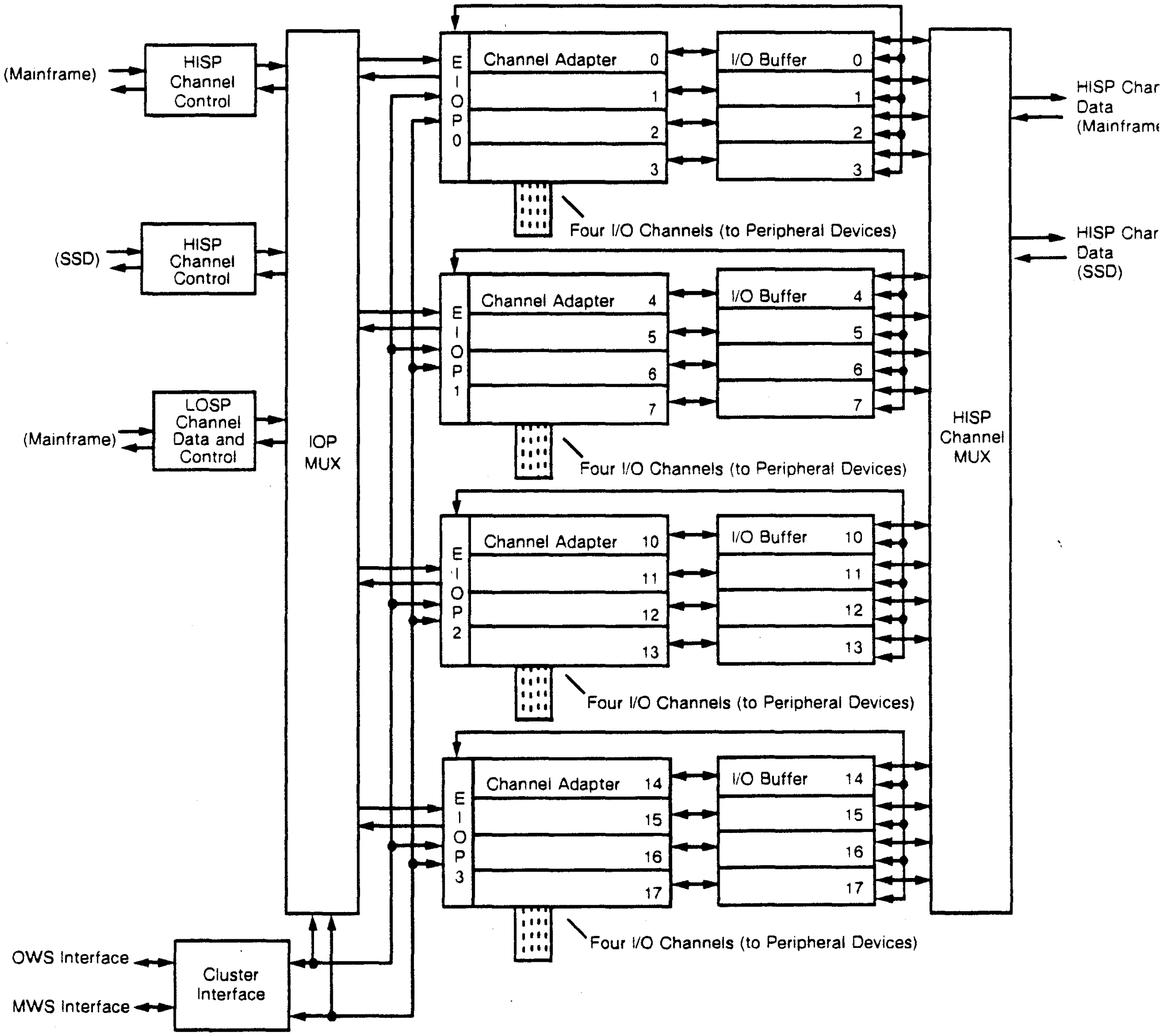


Figure 1-2. I/O Cluster Block Diagram

The IOPs, LOSP channel pair, and CIN control the cluster. The IOPs provide internal control for the cluster. The LOSP channel pair allows the cluster to exchange control information with the mainframe. The CIN enables the workstations to monitor and control the cluster.

The I/O buffers, HISP channels, and channel adapters provide the paths for data transfers between the mainframe or SSD and peripheral devices. For example, consider a data transfer from the mainframe to a disk drive. The mainframe transfers the data through a HISP channel to the IOS which stores the data in an I/O buffer until the disk drive is ready to receive it. When the drive is ready, the IOS transmits the data to the drive through a channel adapter.

Data transfers in the opposite direction (peripheral device to mainframe or SSD) work similarly. For a transfer from a disk drive to the mainframe, the drive first transmits the data to the IOS through a channel adapter. The IOS temporarily stores the data in an I/O buffer, then transmits the data to the mainframe through a HISP channel.

The following subsections describe each of the components of a cluster.

## I/O Processors

The IOPs (one IOP MUX and four EIOPs) control all data transfers into and out of the cluster. The IOP MUX communicates with the mainframe and controls data transfers to or from the mainframe and SSD. The four EIOPs control data transfers to or from peripheral devices. Each EIOP can communicate with the IOP MUX but not with other EIOPs or with the mainframe.

The IOPs are identical; they have the same architecture and execute the same instruction set. Each IOP is a high-speed 16-bit (1-parcel) computer designed to efficiently control data transfers. Each IOP contains a 64-Kparcel local memory (protected with SECDED logic), a 128-parcel operand register file (parity protected), and 3 programmable registers. Each IOP contains 29 I/O channels; 5 of the channels monitor and control operations within the IOP and 24 channels allow the IOP to communicate with external devices.

Each IOP executes a set of 128 1-parcel and 2-parcel instructions. Ninety-six instructions perform basic operations such as data transfers, arithmetic (two's complement), logical and shift operations, conditional and unconditional jumps, and subroutine calls and exits. Thirty-two instructions, called I/O functions, control and monitor the I/O channels. The ELAN assembler supports all IOP instructions.

Section 2 of this manual describes the I/O processors in more detail. Section 7 of this manual explains the IOP instruction set.

## I/O Buffers

The 16 I/O buffers provide temporary storage for data transferred between the mainframe or SSD and peripheral devices. Each buffer contains 65,536 64-bit words and is protected with SECDED logic. Each buffer can transmit data to or from the mainframe or SSD at speeds of more than 200-Mbytes/s, while simultaneously transmitting data to or from a peripheral device at the same rate. Each buffer is dedicated to one peripheral device, or in the case of mass storage devices, to one group of identical devices.

Each EIOP is dedicated to four of the sixteen I/O buffers. Each EIOP controls all transfers between its buffers and peripheral devices, works with the IOP MUX to control transfers to or from the mainframe or SSD, and can also transfer data between its I/O buffers and its local memory.

Section 3 of this manual describes the I/O buffers in more detail.

## LOSP and HISP Channels

The LOSP and HISP channel pairs allow the cluster to communicate with the mainframe and SSD. The LOSP channel pair transfers control information between the IOP MUX and the mainframe. One HISP channel pair transfers data between the I/O buffers and the mainframe; the second pair transfers data between the I/O buffers and the SSD.

The LOSP channel pair can operate at either 6 Mbytes/s or 50 Mbytes/s. (CRAY Y-MP computer systems currently operate at 6 Mbyte/s; 50 Mbyte/s operation is reserved for future use.) The LOSP channels are 16 bits wide and contain 4 parity bits for error detection. Both channels can operate simultaneously.

The HISP channel pairs can operate at either 100 Mbytes/s or 200 Mbytes/s, depending on the capabilities of the mainframe or SSD. The HISP channels are 72 bits wide (64 data bits and 8 SECDED bits). All four channels can operate simultaneously, but each channel must use a different I/O buffer.

Section 4 of this manual describes the LOSP and HISP channels in more detail.

## Channel Adapters

Channel adapters allow the cluster to communicate with peripheral devices. Several types of channel adapters are available, each allowing the cluster to communicate with a different type of device. Most channel adapters are bidirectional, transferring data both to and from peripheral devices, but some channel adapters can transfer data in only one direction. Table 1-1 shows each channel adapter type and its associated channel adapters.

Table 1-1. Channel Adapters

| Channel Adapter | Peripheral Device |
|---|---|
| CCA-1 | LOSP channel |
| DCA-1 | DD-39, DD-49, DD-40, and DD-41 disk drives |
| DCA-2 | DD-60 and DD-61 disk drives |
| HCA-3 | HIPPI input channel |
| HCA-4 | HIPPI output channel |
| TCA-1 | IBM block multiplexer channel |

Each channel adapter corresponds to one I/O buffer. During a data transfer from a peripheral device, the channel adapter converts the input data from the device's format to 64-bit words, generates SECDED bits, and then transmits the data and SECDED bits to the I/O buffer. During a data transfer to an external device, the channel adapter receives 64-bit data words (plus SECDED bits) from the I/O buffer, converts the data to the appropriate format for that device, and then transmits the data to the device.

Each EIOP controls four channel adapters. The EIOPs control and monitor all data transfers between the I/O buffers and peripheral devices. In addition, the EIOPs can transfer data from local memory to peripheral devices.

Section 5 of this manual describes channel adapters in more detail.

### Cluster Interface

The CIN enables the cluster to communicate with the WINs, which in turn enables the cluster to communicate with the workstations. The CIN operates in conjunction with the WIN. The operation of the CIN is transparent to programmers and workstation operators; there are no I/O processor instructions or workstation commands that affect a CIN without affecting a WIN. Because of the relationship between the CIN and WIN, this manual describes the CIN and WIN together in Section 6 of this manual.

## WORKSTATION INTERFACES

The two WINs work with the CINs in each cluster to enable communication with the workstations. Each WIN has a 6-Mbyte/s channel pair; this pair consists of an input and an output channel. Each channel contains 16 data bits and 4 parity bits for data error detection.

The workstations control the entire IOS through the WINs. Each workstation can send WIN commands that affect the entire IOS, a single cluster, or a single I/O processor. The workstations can master clear the entire IOS, an individual cluster, or an individual I/O processor. They can also transfer data to or from any IOP, deadstart an IOP, and monitor IOPs for errors.

The IOPs can send requests to the WINs to transfer data to or from a workstation. However, a workstation receiving a request must send a specific command to the requesting IOP before the transfer can begin.

Section 6 of this manual describes the CINs and WINs in more detail.

# IOS MODEL E SPECIFICATIONS

## General Specifications

| | |
|---|---|
| I/O clusters | 1 to 8 (1 per CPU) |
| Workstation interfaces | 2 |
| Clock speed | 160 MHz (6.25 ns) |

## I/O Cluster

| | |
|---|---|
| I/O processors | |
| Multiplexer I/O processor (IOP MUX) | 1 |
| Auxiliary I/O processors (EIOPs) | 4 |
| I/O buffers | 16 |
| I/O channels | |
| Low-speed channel pair | 1 |
| High-speed channel pairs | 2 |
| Channel adapters | 16 |

## I/O Processor

| | |
|---|---|
| Word width | 16 bits (1 parcel) |
| Instruction set | |
| 1-parcel instructions | 114 |
| 2-parcel instructions | 14 |
| Instruction stack | |
| Size | 32 parcels |
| Data protection | parity |
| Program exit stack | |
| Size | 32 parcels |
| Data protection | parity |
| Real-time clock | |
| Size | 18 bits |
| Range | 0 to 159,999 CPs |
| Maximum interval | 1 ms |
| Programmable registers | |
| Accumulator | 16 bits plus carry bit |
| B register | 9 bits |
| Base register | 7 bits |
| Operand registers | 128×16 bits |
| Functional Units | |
| Adder (two's complement) | 1 |
| Shifter (circular and end-off) | 1 |
| Logical (AND) | 0 † |

## I/O Processor (continued)

| | |
|---|---|
| Local memory | |
| Size | 64 Kparcels |
| Cycle time | 18.75 ns |
| Data protection | SECDED |
| I/O Channels | |
| Internal | 5 |
| External | 24 |

## I/O Buffer

| | |
|---|---|
| Word width | 64 bits |
| Size | 64 Kwords |
| Cycle time | 18.75 ns |
| Data protection | SECDED |
| Maximum data transfer rate | 213 Mbytes/s |

## I/O Channels

| | |
|---|---|
| Low-speed channel pair | |
| Operation | full duplex |
| Channel width | 16 bits (1parcel) |
| Transfer rate | 6 or 50 Mbytes/s ‡ |
| Data protection | parity (4 bits/parcel) |
| High-speed channel pair | |
| Operation | full duplex |
| Channel width | 64 bits |
| Transfer rate | 100 or 200 Mbytes/s |
| Data protection | SECDED |

## Channel Adapters

| | |
|---|---|
| Word width | |
| I/O buffer side | 64 bits |
| Device side | device dependent |
| Data protection | |
| I/O buffer side | SECDED |
| Device side | device dependent |
| Data transfer rate | device dependent |

† AND operations are performed without a functional unit.

‡ Due to IOP limitations, the low-speed channels are limited to approximately 20 Mbytes/s.

# CONTENTS

2. I/O PROCESSOR (continued)

FIGURES

TABLES

# 2. I/O PROCESSOR

Each cluster in the I/O subsystem (IOS) contains a multiplexer I/O processor (IOP MUX) and four auxiliary I/O processors (EIOPs). All five IOPs are identical; they contain the same components and execute the same instruction set. The only differences between the IOP MUX and the EIOPs are the devices they control. The IOP MUX controls the low-speed (LOSP) and high-speed (HISP) channels that connect the IOS to the mainframe and the SSD solid-state storage device. The EIOPs control the channel adapters that connect the IOS to peripheral devices such as disk drives, tape drives, and communication channels.

Each IOP is a high-speed 16-bit (1-parcel) computer designed to efficiently control data transfers into and out of the IOS. Each IOP executes a set of 128 instructions, including 114 1-parcel and 14 2-parcel instructions. Refer to Section 7, "I/O Processor Instruction Set," for detailed information about each instruction.

Each IOP contains four major sections:

- The control section fetches and decodes instructions and controls branches, interrupts, and subroutine calls and returns. Its major components are the instruction stack, program address (P) register, next instruction parcel (NIP) register, current instruction parcel (CIP) register, exit stack, and real-time clock (RTC).
- The computation section provides temporary data storage and performs logical, arithmetic, and shift operations. Its major components are the accumulator, add and shift functional units, operand register file, B register, and base register.
- The local memory section contains 64 Kparcels of random access memory which is protected with single-error correction/double-error detection (SECDED) logic.
- The I/O section contains 29 I/O channels. Five internal channels monitor and control operations within the IOP. Twenty-four external channels provide communication with other components of the IOS and with external devices.

Figure 2-1 shows the organization of an IOP. The remainder of this section provides details on each section of the IOP.

## CONTROL SECTION

The control section controls the flow of instructions in the IOP. It fetches instructions from local memory and stores them temporarily in the instruction stack. It then reads each instruction from the stack and issues the instruction when the needed resources

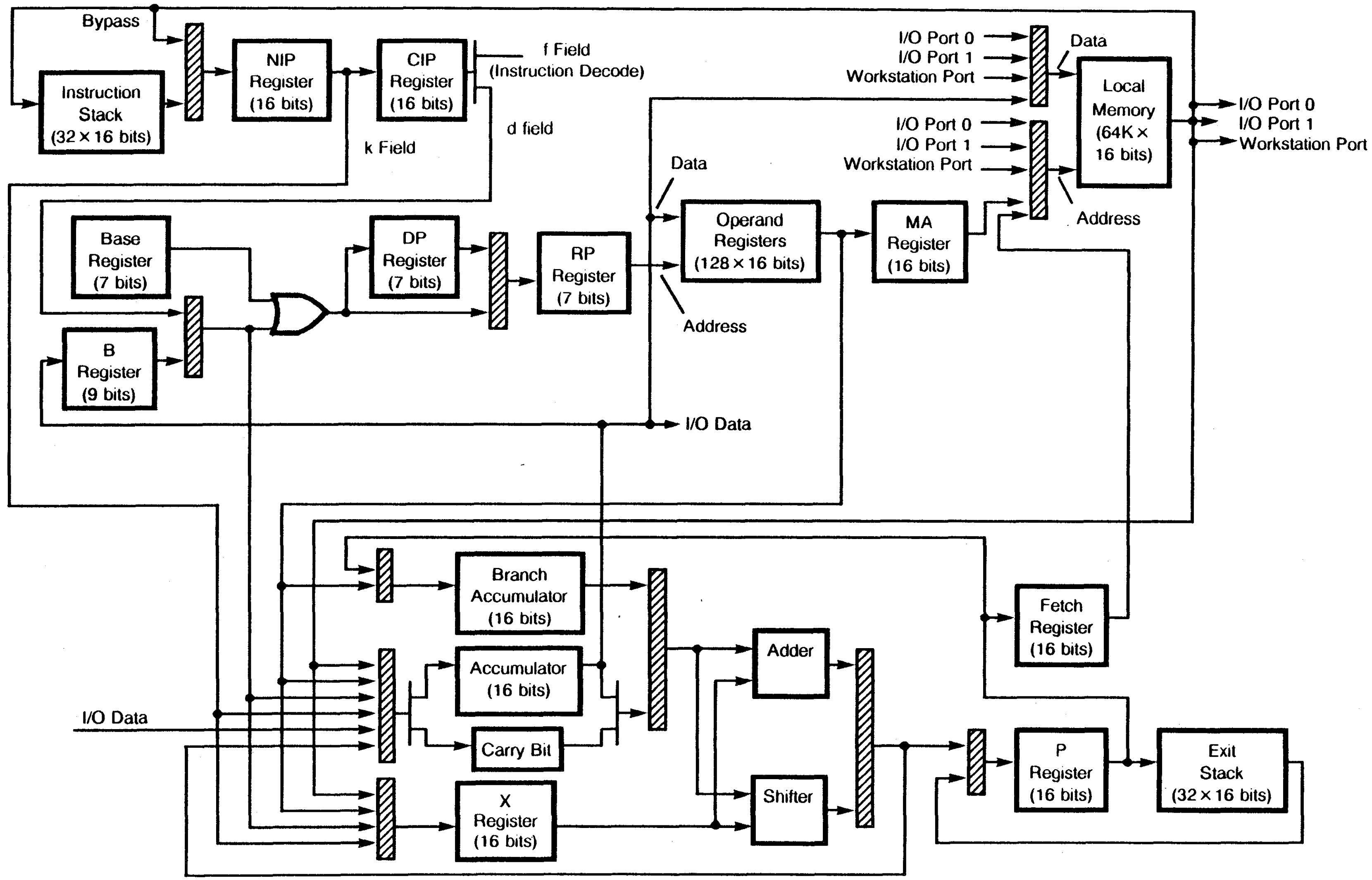


Figure 2-1. IOP Block Diagram

(such as registers and functional units) are available. The control section also controls branches, subroutine calls and exits, and interrupts.

The following subsections describe the components of the IOP control section.

## Instruction Stack

The instruction stack is a 32-parcel buffer that stores instructions temporarily before execution. Instructions are transferred from local memory to the instruction stack and then from the stack to the NIP register.

If a program jumps to an instruction that is already in the stack, the instruction (and possibly subsequent instructions) can be read directly from the stack and does not have to be fetched from local memory. The instruction stack eliminates the delay required to read local memory and eliminates one potential source of local memory access conflict.

A bypass feature allows an instruction to be transferred directly from local memory to the NIP during the same clock period (CP) it is transferred to the instruction stack. The bypass feature is used whenever the NIP register is waiting for an instruction. The bypass transfers an instruction 1 to 3 CPs faster than when it sequentially transfers the instruction to the instruction buffer and then to the NIP register.

The 32 parcels of the instruction stack are arranged in a circular manner. After interrupts, exit instructions, and most jump instructions, the first instruction fetched from local memory is loaded into location 0 of the stack. Assuming only 1-parcel instructions are fetched, the next instruction is loaded into location 1, and so on until all 32 locations are filled. After the entire stack is full, the next instruction overwrites location 0 and subsequent instructions overwrite the remaining parcels in sequence. This process continues until an interrupt, exit instruction, or jump instruction occurs.

If an interrupt or exit instruction occurs, the instruction stack is reloaded beginning at location 0. Jump (and return jump) instructions affect the instruction stack differently, depending on the type of jump instruction and the destination. Absolute jump instructions (those that jump to an absolute memory location – instructions 074 through 077 and 120 through 127) cause the instruction stack to be reloaded beginning at location 0.

Relative jump instructions, also called branch instructions, (instructions 070 through 073 and 100 through 117) jump to locations relative to the location of the jump instruction. The destination of a relative jump instruction may already be in the instruction stack, in which case an in-stack branch occurs. If the destination instruction must be fetched from local memory, an out-of-stack branch occurs. Out-of-stack branches cause the instruction stack to be reloaded beginning at location 0.

Control logic associated with the instruction stack includes a write pointer, a read pointer, and branch control logic. The write and read pointers are initialized to 0 each time an interrupt, exit instruction, or out-of-stack branch occurs. The write pointer increments after each instruction parcel is written to the stack from local memory; the read pointer increments after each parcel is read from the stack to the NIP register. Because instructions can often be written to the stack faster than they can be read from the stack, the write pointer often increments faster than the read pointer. The control logic allows the write pointer to be up to 12 locations ahead of the read pointer; when this condition is reached, the write pointer is prevented from further incrementing until the read pointer increments.

The branch control logic uses the read and write pointers to determine if a jump instruction requires an in-stack or out-of-stack branch. If an in-stack branch occurs, the read pointer is changed to the new location and the next instruction is read from that location. If an out-of-stack branch occurs, both pointers are initialized to 0 and the next instruction is fetched from local memory.

Several factors determine whether a jump instruction causes an in-stack or out-of-stack branch:

- Instructions that jump to an absolute local memory address (instructions 074 through 077 and 120 through 137) always cause an out-of-stack branch.

- Instructions that jump forward or backward relative to the current program address (instructions 070 through 073 and 100 through 117) may cause either an in-stack branch or out-of-stack branch, depending on several factors:

    - A forward branch greater than 12 or a backward branch greater than 20 always causes an out-of-stack branch.

    - A forward branch of 12 or less causes an in-stack branch if the write pointer has advanced at least to the branch destination. (The write pointer attempts to remain 12 locations ahead of the read pointer, but cannot always do so.)

    - A backward branch of 20 or less causes an in-stack branch unless the branch destination in the instruction stack is not valid. An invalid destination can occur following a stack initialization (due to an interrupt, exit instruction, or out-of-stack branch) before the first 20 stack locations are loaded.

A parity generation and checking circuit protects the instruction stack from data errors. Two parity bits are stored with each parcel written to the instruction stack; the lower parity bit protects bits $2^0$ through $2^7$, and the upper parity bit protects bits $2^8$ through $2^{15}$. When a parcel is read from the stack, new parity bits are generated and compared with the parity bits originally stored with the parcel. If the two pairs of parity bits are unequal, a parity error has occurred.

A parity error is reported to the local memory error channel and to the workstation interface. Under normal operating conditions, a parity error also causes the IOP to stop issuing instructions. However, for maintenance purposes, a workstation can command an IOP to disregard instruction stack parity errors and continue program execution. For maintenance purposes, the base register channel can force a parity error to occur whenever an instruction is read from the instruction stack. Refer to the "Channel 7- Base Register Select (BRS)" subsection in this section for more information on forcing parity errors to occur.

## Next Instruction Parcel Register

The 16-bit NIP register receives the instruction parcel from the instruction stack or from local memory during a bypass operation and partially decodes it. The parcel remains in the NIP register until the CIP register is ready to receive it.

For 2-parcel instructions, the NIP register transmits only the first parcel to the CIP register; the NIP register transmits the second parcel directly to its destination.

## Current Instruction Parcel Register

The CIP register receives the instruction from the NIP register, decodes it, then holds the instruction until all conditions required to issue the instruction are met. (Refer to Section 7 of this manual, "I/O Processor Instruction Set," for a list of the hold issue conditions for each instruction.) When the issue conditions are met, the CIP register generates all control signals needed to execute the instruction.

## Program Address Register

The 16-bit P register holds the local memory address of the instruction waiting to issue in the CIP register. The P register is updated after each instruction is issued.

During normal program execution, the P register increments by 1 or 2 (depending on the instruction length) after each instruction issues. Following a branch instruction, interrupt, or subroutine call or exit, the P register is loaded with the address of the next instruction to be issued.

## Fetch Register

The 16-bit fetch register holds the local memory address of the next instruction parcel to be fetched from local memory. Following branch instructions, exit instructions, and interrupts, the fetch pointer is loaded from the P register. As each instruction parcel is fetched, the fetch pointer increments.

## Program Exit Stack

The program exit stack is a 32-parcel register file that stores addresses. The stack locations are numbered from 0 to 31.

Stack location 0 stores the address of the interrupt handler (the subroutine called when an interrupt occurs). Location 0 is written only under program control; it is read automatically whenever an interrupt occurs.

Stack locations 1 through 31 are organized as a last-in-first-out buffer. These locations store return addresses for nested subroutine calls and interrupts. An address is automatically written to one of these locations each time a return jump instruction or an interrupt occurs. An address is automatically read from one of these locations each time an exit instruction occurs. Locations 1 through 31 can also be written and read under program control.

The exit stack (E) pointer is a 5-bit register that selects one location in the exit stack. The E pointer increments or decrements automatically whenever a return jump instruction, exit instruction, or interrupt occurs. It can also be incremented, decremented, and read under program control.

By using a program to transfer the stack contents to and from local memory, the logical size of the stack can be increased beyond its physical size. For details on programming the exit stack, refer to "Program Exit Stack Channel" later in this section.

A parity generation and checking circuit protects the program exit stack from data errors. Two parity bits are stored with each parcel written to the exit stack; the lower parity bit protects bits $2^0$ through $2^7$, and the upper parity bit protects bits $2^8$ through $2^{15}$. When a parcel is read from the stack, new parity bits are generated and compared with the parity bits originally stored with the parcel. If the two pairs of parity bits are unequal, a parity error has occurred.

A parity error is reported to the program exit stack channel and to the workstation interface. Under normal operating conditions, a parity error also causes the IOP to stop issuing instructions. However, for maintenance purposes, a workstation can command an IOP to disregard exit stack parity errors and continue program execution. For maintenance purposes, the program exit stack channel can force a parity error to occur whenever the exit stack is read. Refer to "Channel 2 - Program Exit Stack (PXS)," later in this section, for more information on forcing parity errors to occur.

The operation of the exit stack during subroutine calls and exits and during interrupts is explained in the following subsections.

### Subroutine Calls

During a subroutine call (return jump instruction), the E register first increments. The contents of the P register are then written to the exit stack at the new location designated by the E pointer. If the E pointer is equal to 29 ($35_8$) when a subroutine call occurs, the overflow flag sets and an interrupt request is generated. If the E pointer is equal to 31 ($37_8$) when a subroutine call issues, the E pointer does not increment and the P register is written to location 31.

### Subroutine Exits

During a subroutine exit (exit instruction), the exit stack location designated by the E pointer is written to the P register; the E pointer then decrements. If the E pointer is at 0 when the exit instruction issues, the underflow flag sets, an interrupt request is generated, and the E pointer remains at 0.

### Interrupts

When an interrupt request is granted, the contents of exit stack location 0 are written to the P register. The E pointer is incremented and the previous contents of the P register is written to the exit stack at the location now designated by the E pointer. If the E pointer is equal to 29 ($35_8$) when an interrupt occurs, the overflow flag is set and an exit stack interrupt request is generated. If the E pointer is equal to 31 ($37_8$) when an interrupt occurs, the E pointer does not increment and the P register is written to location 31.

## Real-time Clock

The real-time clock (RTC) is an 18-bit counter that increments once each CP. Its primary purposes are to generate periodic interrupt requests and to time intervals between events. Refer to "Real-time Clock Channel" later in this section for more information on programming the RTC to perform these functions.

The RTC increments to a maximum value of 159,999 ($470377_8$); during the following CP it resets to 0. Because the CP is 6.25 ns, the RTC resets to 0 at 1-millisecond intervals (6.25 ns × 160,000). Each time the RTC resets, it generates an interrupt request. The RTC runs continuously; it cannot be stopped and its count cannot be changed.

# COMPUTATION SECTION

The computation section of an IOP provides temporary data storage and performs logical, arithmetic, and shift operations. The computation section also provides addresses for memory reference instructions. The major components of the computation section are described in the following subsections.

## Registers

The major registers in the IOP computation section are described in the following subsections.

### Accumulator

The 16-bit accumulator is the primary data register in an IOP. All instructions that transfer data within an IOP use the accumulator as the data source, destination, or both. All instructions that perform logical, shift, and arithmetic functions use the accumulator as an operand and as the destination for the result. The following IOP components are sources and destinations for accumulator data:

| Source | Destination |
|---|---|
| B register | B register |
| Operand registers | Operand registers |
| Functional units | Functional units |
| Local memory | Local memory |
| I/O channels | I/O channels |
| Instruction *d* field | |
| Instruction *k* field | |

Some conditional jump instructions test the accumulator to determine if a jump should be made.

### Carry Bit

The carry bit is used for the following types of instructions:

- Add and shift instructions use the carry bit as bit $2^{16}$ of the accumulator. Refer to "Adder" and "Shifter" in this section for details.

- All instructions that write to the accumulator, except add and shift instructions, clear the carry bit.

- Instructions 040 through 043 transfer the contents of a channel busy or done flag to the carry bit. (Refer to the subsection "I/O Channels" in this section for more information on the busy and done flags.)

- Some conditional jump instructions test the carry flag to determine if a jump should be made.

### Branch Accumulator

The 16-bit branch accumulator receives data from the P register during relative branch instructions. The branch accumulator transmits the P register data to the adder, which computes the branch address.

### Addend (X) Register

The 16-bit addend (X) register supplies an operand to the add functional unit and the shift count to the shift functional unit. The X register receives data from the *d* or *k* field of the instruction, the B register, an operand register, or local memory. The source of X register data is determined by the instruction being executed.

### B Register

The 9-bit B register provides the following data for different instructions:

- A functional unit operand
- A pointer to one of the 128 operand registers
- An I/O channel designator

The B register receives data from the accumulator. It can also be incremented or decremented by 1, or incremented by the contents of the accumulator.

Instructions that increment or decrement the B register use the adder to compute the result. The adder transmits the result to the accumulator, then the accumulator transmits the result to the B register. Therefore, these instructions change the contents of the accumulator as well as the contents of the B register.

## Operand Registers

The 128 operand registers store data and jump addresses and provide addresses for all local memory reference instructions. Each 16-bit operand register has a cycle time of 1 CP.

The *d* field of the instruction or the B register selects one of the 128 operand registers to transmit or receive data. In some cases, the contents of the base register modify the selection made by the *d* field or B register. Refer to "Base Register" in this section for more information on operand register addressing.

An operand register can transmit data to the following registers:

- Accumulator
- X register
- Memory address (MA) register

Operand registers receive data from the accumulator. They can also be incremented or decremented by 1 or incremented by the contents of the accumulator.

Instructions that increment or decrement an operand register use the adder to compute the result. The adder transmits the result to the accumulator, then the accumulator transmits the result to the operand register. Therefore, these instructions change the contents of the accumulator as well as the contents of an operand register.

A parity generation and checking circuit protects the operand registers from data errors. Two parity bits are stored with each parcel written to the operand registers; the lower parity bit protects bits $2^0$ through $2^7$ and the upper parity bit protects bits $2^8$ through $2^{15}$.

When a parcel is read from the stack, new parity bits are generated and compared with the parity bits originally stored with the parcel. If the two pairs of parity bits are unequal, a parity error has occurred.

A parity error is reported to the local memory error channel and to the workstation interface. For maintenance purposes, the local memory error channel can force a parity error to occur whenever an operand register is read. Refer to "Channel 3 - Local Memory Error (LME)" in this section for more information on how parity errors are handled.

## Base Register

The 7-bit base register works with the *d* field of the instruction or with the B register to provide operand register addressing. Bit $2^8$ of the *d* field determines how the base register is used. If bit $2^8$ is 0, the base register is disregarded and bits $2^0$ through $2^6$ of the *d* field or B register provide the operand register address. If bit $2^8$ of the *d* field is 1, bits $2^0$ through $2^6$ of the *d* field is ORed with the corresponding bits of the base register; the result provides the operand register address.

The base register is loaded under program control. Refer to "Channel 7-Base Register Select (BRS)," later in this section, for more information.

### Reference Pointer Register

The 7-bit reference pointer (RP) register selects one of the 128 operand registers to receive or transmit data. The RP register receives data from the *d* field (instructions 020 through 037) or B register bits $2^0$ through $2^6$ (instructions 060 through 067). If *d* field bit $2^8$ is 1, the RP register receives the *d* field or B register contents ORed with the base register.

### Destination Pointer Register

The 7-bit destination pointer (DP) register is used only by instructions 025 through 027 and 065 through 067. These instructions read data from an operand register, modify the data, then write the modified data back to the same operand register.

The RP and DP registers work together during these instructions. During the read portion of an instruction, the DP and RP registers receive the operand register address simultaneously. The RP register uses the address immediately to select the register to be read. The DP register latches the operand register address until the write portion of the instruction, then transmits the address to the RP register. The RP register then uses the address to select the register to be written to.

### Memory Address Register

The 16-bit memory address (MA) register provides the local memory address for all instructions that access local memory. The MA register receives the local memory address from an operand register.

## Functional Units

The add functional unit provides integer addition and subtraction. The shift functional unit provides circular and end-off shifts.

**Note:** The IOP computation section also computes logical products. However, logical products do not require a functional unit. Logical products are formed when data is written to the accumulator; no additional time is required to compute the product.

### Adder

The adder performs two's-complement addition and subtraction. The first operand is obtained from the accumulator and the second operand is obtained from the X register. The result is transmitted to the accumulator. Some instructions also transmit the result to the B register, an operand register, or local memory.

The adder treats the carry bit as bit $2^{16}$ of the accumulator; therefore, the adder uses one 17-bit operand (accumulator and carry bit), one 16-bit operand (X register), and produces a 17-bit result (accumulator and carry bit). Overflows to bit $2^{17}$ are ignored.

The adder computes destination addresses for jump instructions. For these instructions, the first operand is obtained from the branch accumulator, and the second operand is obtained from the X register. The result is transmitted to the P register and does not affect the accumulator or carry bit.

The adder requires 2 CPs to compute a result. An additional CP is required to write the result into the destination register.

### Shifter

The shifter performs circular and end-off shifts. The operand is obtained from the accumulator and the result is transmitted back to the accumulator. The shifter treats the carry bit as bit $2^{16}$ of the accumulator; therefore, the functional unit uses a 17-bit operand and produces a 17-bit result.

The shift count is obtained from the X register. Only bits $2^0$ through $2^4$ of the X register are used by the shifter; bits $2^5$ through $2^{15}$ are disregarded. The maximum shift count is 31. For end-off shifts, a shift count greater than 16 clears the accumulator and carry bit regardless of their previous values.

The shifter requires 2 CPs to compute a result. An additional CP is required to write the result into the accumulator.

# LOCAL MEMORY

Each IOP has a random-access local memory for program and data storage. Local memory consists of 64 Kparcels (65,536 parcels) organized as a single memory bank. Local memory has a cycle time of 3 CPs and is protected with SECDED logic.

Local memory transmits data to the instruction stack, accumulator and external I/O channels. Local memory receives data from the accumulator, addend (X) register, and external I/O channels.

The following subsections explain how the different components of the IOS access local memory and how local memory errors are detected and corrected.

## Local Memory Access

Three areas of an IOP have access to local memory:

- The instruction stack fetches instructions from local memory. This type of access is called a fetch reference.
- Instructions can read or write data to local memory. This type of access is called a memory access (MA) reference.
- Some external I/O channels can read and write data to or from local memory. This type of access is called an I/O reference.

One local memory reference (read or write) can occur every 3 CPs. The 3-CP interval is called a memory cycle. If more than one memory reference is requested at the same time, conflict resolution logic allows one reference to proceed and forces the other references to wait until local memory is not busy.

The following priority scheme resolves local memory conflicts:

- Reference requests are divided into two groups. Fetch and MA references comprise one group; I/O references comprise the other group. Priority alternates between MA/fetch requests and I/O requests each memory cycle.

- MA references always have higher priority than fetch references.

- I/O references use one of three memory ports, depending on the channel number. The workstation channels (22 through 25) use the workstation port. Channels 20, 21, 26, 27, 30, 31, 34, and 35 use port 0. Channels 32, 33, 36, and 37 use port 1.

  The workstation port has the highest priority, followed by port 1, and finally by port 0. A port is prevented from making two successive memory requests if another port has a request pending. For example, if the workstation port is making a request and port 1 has a request pending, the workstation port cannot make another request until port 1's memory request is granted.

- Within each port, odd-numbered channels (local memory read references) have priority over even-numbered channels (write reference). However, during the memory cycle following a read reference, write references have priority.

- The priority among even-numbered I/O channels within a port rotates each CP (not each memory cycle). For example, in port 0, channel 20 has priority for one CP, followed by channels 26, 30, and 34. Priority then returns to channel 20 the following CP.

## Error Detection and Correction

SECDED logic protects local memory from data errors. Single-bit errors can be detected and corrected. Double-bit errors can be detected, but not corrected. Errors involving more than 2 bits cannot be reliably detected.

The SECDED scheme works as follows. When a parcel (bits $2^0$ through $2^{15}$) is written to memory, 6 check bits are generated and stored in memory with the data parcel. (The check bits are numbered 0 through 5 and are stored as data bits $2^{16}$ through $2^{21}$.) When the word is read from memory, a set of check bits is again generated and compared with the original check bits using an exclusive OR. The resulting comparison is called a syndrome. If no memory error occurred, the two sets of check bits are identical and all the bits in the syndrome are 0.

If there are one or more ones in the syndrome, a memory error occurred. The type of memory error (single-bit or double-bit) can be determined by interpreting the syndrome. If a single-bit error occurred, the syndrome indicates the bit in error and the SECDED logic toggles the incorrect bit to its correct value. If a double-bit error occurred, the syndrome indicates that there was an error, but it cannot locate the incorrect bits. Errors involving more than 2 bits produce unpredictable results. In some cases the errors produce unique syndromes that can be detected by the SECDED logic. In other cases the syndrome appears to be a no-error condition or a single- or double-bit error.

Table 2-1 lists the data bits that generate each bit in the check byte. All data bits marked with an X contribute to the corresponding check bit. The parity of all such data bits determines the state of the check bit. If the parity is even, the check bit is cleared. If it is odd, the check bit is set. For example, the data bits that make up check bit 0 are bits $2^0$, $2^1$, $2^2$, $2^4$, $2^5$, $2^7$, $2^{11}$, $2^{12}$, and $2^{14}$. If an even number of these bits is equal to 1, check bit 0 is cleared; otherwise check bit 0 is set.

Table 2-1. Local Memory Check Bits

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

When a single- or double-bit error occurs, the error is reported to the local memory error channel and to the workstation interface. A double-bit error also causes an interrupt request. A single-bit error may or may not cause an interrupt request, depending on the the state of a mode bit controlled by the local memory error channel. For maintenance purposes, the local memory error channel can also disable the writing of check bits to local memory. Refer to "Channel 3 - Local Memory Error (LME)," in this section, for more information on how local memory errors are handled.

## I/O CHANNELS

I/O channels allow an IOP to communicate with its own components, with other components in the IOS, and with peripheral devices. The channels can be divided into two categories, internal and external. Internal channels monitor and control operations within the IOP; external channels provide communications with other parts of the IOS (I/O buffer, other IOPs), with the mainframe and SSD, and with external devices such as workstations and mass storage devices.

Channel functions (IOP instructions 140 through 177) control the I/O channels. Bits $2^0$ through $2^3$ of the *d* field (instructions 140 through 147) or B register (instructions 160 through 177) select a channel. Bits $2^0$ through $2^3$ of the *f* field specify a channel operation, such as reading or writing to a register or starting a data transfer.

Each I/O channel, except the base select register channel, has a busy and a done flag that a program can read. For the internal channels, these flags reflect specific hardware conditions. For external channels, the busy flag indicates that a data transfer is in progress, and the done flag indicates that the transfer is complete.

Each I/O channel, except the I/O request and base select register channels, can generate interrupt requests. Internal channels generate interrupt requests when specific conditions occur. External channels generate interrupt requests when the done flag sets at the completion of a data transfer, or when an error occurs while a channel is active..

Each channel with an interrupt capability has a programmable interrupt enable flag that enables or disables interrupt requests. A programmable system interrupt enable flag can disable interrupt requests from all I/O channels, except the program exit stack channel. For each channel except the program exit stack channel, both the channel interrupt enable flag and the system enable flag must be set before interrupts can occur. Program exit stack interrupts can occur if the channel interrupt enable flag is set, regardless of the state of the system interrupt enable flag.

Table 2-2 shows the purpose of each channel in the IOP MUX and EIOPs. Channels 0, 2, 3, 4, and 7 are the internal channels. They perform the same functions for all IOPs. Channels 10 through 37 are the external channels. These channels perform different functions on the IOP MUX than they do on the EIOPs.

Table 2-2. IOP Channels

| Channel Number | IOP MUX | EIOPs |
|---|---|---|
| 0 | I/O request | I/O request |
| 2 | Program exit stack | Program exit stack |
| 3 | Local memory error | Local memory error |
| 4 | Real-time clock | Real-time clock |
| 7 | Base select register | Base select register |
| 10 and 12 | HISP 0 input (mainframe) | Not used |
| 11 and 13 | HISP 0 output (mainframe) | Not used |
| 14 and 16 | HISP 1 input (SSD) | Not used |
| 15 and 17 | HISP 1 output (SSD) | Not used |
| 20 and 21 | LOSP (mainframe) | IOP MUX |
| 22 and 23 | OWS | OWS |
| 24 and 25 | MWS | MWS |
| 26 and 27 | Not used | I/O buffer |
| 30 and 31 | EIOP 0 | Channel adapter N + 0 † |
| 32 and 33 | EIOP 1 | Channel adapter N + 1 † |
| 34 and 35 | EIOP 2 | Channel adapter N + 2 † |
| 36 and 37 | EIOP 3 | Channel adapter N + 3 † |

† N equals 0 for EIOP 0, 1 for EIOP 1, 2 for EIOP 2, and 3 for EIOP 3.

The following subsections describe the internal and external channels.

## Internal Channels

The internal channels monitor and control operations within the IOP. Most of the channels have busy and done flags that indicate specific conditions and that can generate interrupt requests.

### Channel 0 - I/O Request

The I/O request (IOR) channel reads interrupt requests from all other I/O channels and indicates the source of interrupt requests. Table 2-3 lists the IOR channel functions.

Table 2-3. IOR Channel Functions

| Function | Description |
|---|---|
| IOR:10 | Read interrupt number |
| IOR:11 | Read interrupts for channels 0 through 17 |
| IOR:12 | Read interrupts for channels 20 through 37 |

The IOR busy flag indicates the current state of the system interrupt flag. The done flag indicates the state of the system interrupt flag when the most recent disable system interrupt (002) instruction issued. The IOR channel does not generate interrupt requests.

The following subsections explain each IOR channel function.

#### IOR:10 - Read Interrupt Number

This function transfers the number of the highest-priority channel requesting an interrupt to the accumulator. Channel priority is based on channel number; the lower the number, the higher the priority. If no channels are requesting an interrupt when this function is issued, a value of 0 is transferred to the accumulator.

#### IOR:11 - Read Interrupts for Channels 0 through 17

This function transfers the interrupt request flags for channels 0 through 17 to the accumulator. Figure 2-2 shows the format of the interrupt request flags in the accumulator. Bits $2^0$, $2^1$, and $2^5$ through $2^7$ are 0; these bits correspond to unassigned channels or channels without interrupt capability.

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ch. 17 | Ch. 16 | Ch. 15 | Ch. 14 | Ch. 13 | Ch. 12 | Ch. 11 | Ch. 10 | 0 | 0 | 0 | Ch. 4 | Ch. 3 | Ch. 2 | 0 | 0 |

Note: EIOP channels 10 through 17 are not used.

Figure 2-2. IOR:11 Status Parcel

IOR:12 - Read Interrupts for Channels 20 through 37

This function transfers the interrupt request flags for channels 20 through 37 to the accumulator. Figure 2-3 shows the format of the interrupt request flags in the accumulator.

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ch. 37 | Ch. 36 | Ch. 35 | Ch. 34 | Ch. 33 | Ch. 32 | Ch. 31 | Ch. 30 | Ch. 27 | Ch. 26 | Ch. 25 | Ch. 24 | Ch. 23 | Ch. 22 | Ch. 21 | Ch. 20 |

Note: IOP MUX channels 26 and 27 are not used.

Figure 2-3. IOR:12 Status Parcel

## Channel 2 – Program Exit Stack (PXS)

The PXS channel monitors and controls the program exit stack. Table 2-4 shows the PXS channel functions.

Table 2-4. PXS Channel Functions

| Function | Description |
|---|---|
| PXS:0 | Clear channel |
| PXS:1 | Set E pointer to 1 |
| PXS:6 | Clear interrupt enable flag |
| PXS:7 | Set interrupt enable flag |
| PXS:10 | Read status |
| PXS:11 | Read location (E) |
| PXS:14 | Increment E pointer |
| PXS:15 | Decrement E pointer |
| PXS:16 | Write location (E) |
| PXS:17 | Write modes |

The PXS busy flag is always clear. The done flag is set if the exit stack overflow, underflow, or parity error flag is set. The PXS channel generates an interrupt request when an underflow, overflow, or parity error occurs.

**Note:** Unlike all other channels, the PXS channel is not affected by the system interrupt enable flag. The PXS channel can generate interrupts if the channel interrupt enable flag is set, regardless of the state of the system interrupt enable flag.

The following subsections explain each PXS channel function.

## PXS:0 - Clear Channel

This function clears the overflow, underflow, and parity error flags. Clearing these flags clears the done flag.

## PXS:1 - Set E Pointer to 1

This function sets the E pointer to 1, the lowest exit stack location available for storing return addresses.

## PXS:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag. A master clear command also clears this flag.

## PXS:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag.

## PXS:10 - Read Status

This function transfers the contents of the E pointer and the states of the underflow, overflow, and parity error flags to the accumulator. Figure 2-4 shows the format of this information in the accumulator.

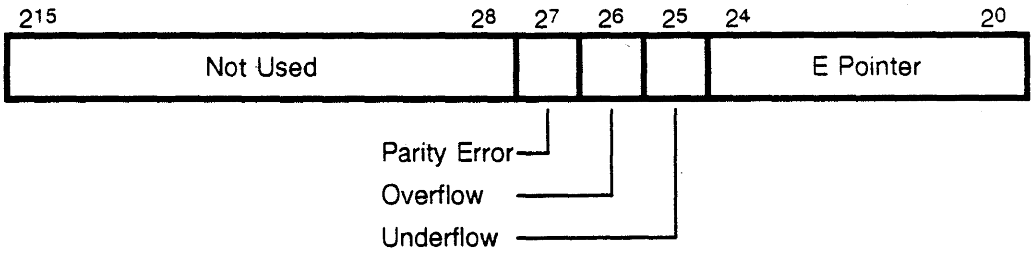


Figure 2-4. PXS:10 Status Parcel

PXS:11 - Read Location (E)

This function transmits the contents of the exit stack location designated by the E pointer to the accumulator.

PXS:14 - Increment E Pointer

This function increments the E pointer. If the E pointer is 29 ($35_8$) when this function issues, the PXS overflow flag sets, the done flag sets, and an interrupt request is generated. If the E pointer is 31 ($37_8$) when this function is issued, the E pointer remains at 31.

PXS:15 - Decrement E Pointer

This function decrements the E pointer. If the E pointer is 0 when this function issues, the E pointer remains 0, the PXS underflow flag sets, the PXS done flag sets, and an interrupt request is generated.

PXS:16 - Write Location (E)

This function transmits the contents of the accumulator to the exit stack location designated by the E pointer.

PXS:17 - Write Modes

This function writes the contents of accumulator bits $2^0$ through $2^2$ to 3 mode bits. These bits are used only for maintenance purposes. Figure 2-5 shows the format of these mode bits in the accumulator.

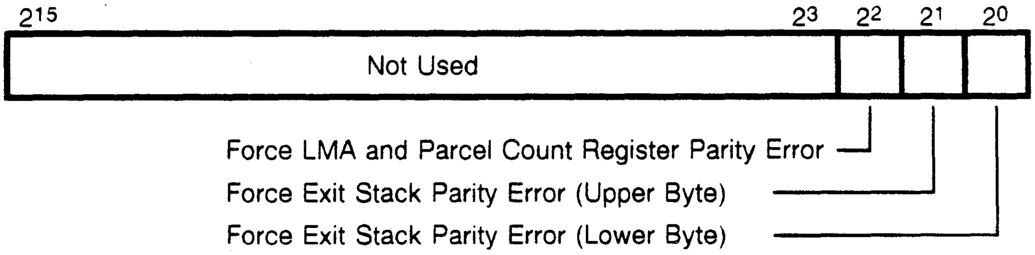

$2^{15}$
$2^3$
$2^2$
$2^1$
$2^0$
Not Used
Force LMA and Parcel Count Register Parity Error
Force Exit Stack Parity Error (Upper Byte)
Force Exit Stack Parity Error (Lower Byte)

Figure 2-5. PXS:17 Control Parcel

Mode bits $2^0$ and $2^1$ affect the operation of the exit stack parity generation logic and are used only for maintenance purposes. When mode bit $2^0$ is set, the parity bit that protects data bits $2^0$ through $2^7$ is inverted before it is stored. When mode bit $2^1$ is set, the parity bit that protects data bits $2^8$ through $2^{15}$ is inverted before it is stored. If either mode bit is set when a parcel is written to the exit stack, a parity error occurs when that parcel is read from the stack.

Mode bit $2^2$ affects the operation of the local memory address (LMA) and parcel count registers associated with I/O channels 10 through 37. (These registers are explained under "External Channels" later in this section.) When mode bit $2^2$ is set, the parity bit that protects each of these registers is inverted before it is stored. If this mode bit is set when data is written to one of these registers, a parity error occurs when that register is read. Once set, a mode bit remains set until cleared by a master clear command or another PXS:17 function.

## Channel 3 - Local Memory Error (LME)

The LME channel monitors local memory SECDED errors, monitors register parity errors (instruction stack, operand register, local memory address registers, and parcel count register), and controls some of the SECDED logic. (Local memory address registers and parcel count registers are discussed in "External Channels" later in this section.) Table 2-5 lists the LME channel functions.

Table 2-5. LME Channel Functions

| Function | Description |
|---|---|
| LME:0 | Clear channel |
| LME:6 | Clear interrupt enable flag |
| LME:7 | Set interrupt enable flag |
| LME:10 | Read syndromes |
| LME:11 | Read processor error flags |
| LME:14 | Write modes |

If a SECDED error or a parity error occurs, the LME busy flag sets and an interrupt request is generated. The done flag is always clear.

### LME:0 - Clear Channel

This function clears the SECDED and parity error flags. Clearing these flags also clears the LME busy flag.

### LME:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag. A master clear command also clears this flag.

### LME:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag.

LME:10 - Read Syndromes

This function reads the correctable and uncorrectable error flags, the processor error flag, and the correctable and uncorrectable error syndromes; the result is transmitted to the accumulator. Figure 2-6 shows the format of this information in the accumulator.

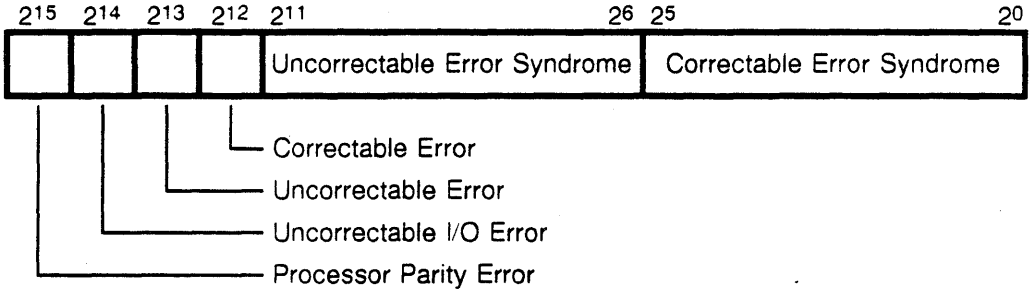


Figure 2-6. LME:10 Status Parcel

If a correctable error occurs during a local memory read operation, bit $2^{12}$ sets and the syndrome is stored in bits $2^0$ through $2^5$. If an uncorrectable error occurs, bit $2^{13}$ sets and the syndrome is stored in bits $2^6$ through $2^{11}$. If the uncorrectable error occurs during a read from local memory to an external I/O channel, bit $2^{14}$ also sets. If a parity error occurs during an instruction stack, local memory address register, parcel count register, or operand register read operation, bit $2^{15}$ sets; function LME:11 provides more information about the location of the parity error.

Function LME:0 or a master clear command clears all the error flags, the syndromes, and the LME busy flag. The first correctable-error syndrome and the first uncorrectable-error syndrome following an LME:0 or master clear are retained; subsequent syndromes are disregarded until the next LME:0 function or master clear command.

LME:11 - Read Processor Error Flags

This function copies all parity error flags, except the exit stack parity error flag, to the accumulator. (Function PXS:10 reads the exit stack error flags.) Figure 2-7 shows the format of the parity error flags in the accumulator.

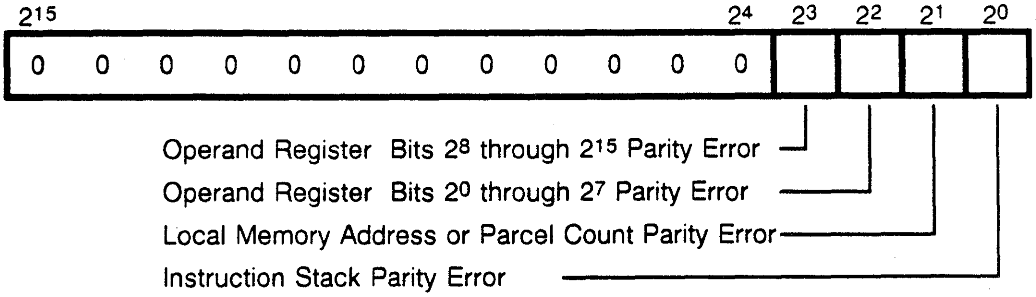

$2^{15}$
$2^4$
$2^3$
$2^2$
$2^1$
$2^0$
0 0 0 0 0 0 0 0 0 0 0 0
Operand Register Bits $2^8$ through $2^{15}$ Parity Error
Operand Register Bits $2^0$ through $2^7$ Parity Error
Local Memory Address or Parcel Count Parity Error
Instruction Stack Parity Error

Figure 2-7. LME:11 Status Parcel

LME:14 - Write Modes

This function writes the contents of accumulator bits $2^0$ through $2^2$ to 3 mode bits; the function is used primarily for maintenance purposes. If one of these bits is set, it remains set until cleared by a master clear command or another LME:14 function.

If bit $2^0$ is set, correctable local memory errors generate interrupt requests. If bit $2^0$ is clear, a correctable error sets the correctable error flag, but does not generate an interrupt request.

If bit $2^1$ is set, the parity bits that protect the operand registers are inverted before they are stored. If bit $2^1$ is set when a parcel is written to an operand register, a parity error occurs when that register is read.

If bit $2^2$ is set, the writing of local memory check bits is disabled; when a parcel is written to local memory, the check bits already stored at the address are not changed.

## Channel 4 – Real-time Clock (RTC)

The RTC channel reads and controls the real-time clock. Table 2-6 shows the RTC channel functions. The RTC busy flag is always set. The RTC done flag sets each time the RTC resets to 0.

Table 2-6. RTC Channel Functions

| Function | Description |
|---|---|
| RTC:0 | Clear done flag |
| RTC:6 | Clear interrupt enable flag |
| RTC:7 | Set interrupt enable flag |
| RTC:10 | Read status |

RTC:0 - Clear Done Flag

This function clears the done flag.

RTC:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag. A master clear command also clears this flag.

RTC:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag.

RTC:10 - Read Status

This function transfers the upper 16 bits of the RTC to the accumulator. (The lower 2 bits of the RTC cannot be read.) Because the RTC generates interrupt requests at 1-millisecond intervals, this function can be used in conjunction with RTC interrupts to determine the interval between two events.

To determine the number of CPs between two events, use the following equation:

$$\text{Time (CPs)} = [(\text{Interrupts} \times 116100_8) + \text{RTC}_{\text{ending}} - \text{RTC}_{\text{beginning}})] \times 4$$

To determine the time in milliseconds between two events, use the following equation:

$$\text{Time (milliseconds)} = \text{Interrupts} + (\text{RTC}_{\text{ending}} - \text{RTC}_{\text{beginning}})/116100_8$$

## Channel 7 - Base Register Select (BRS)

The BRS channel writes the base register and controls the instruction stack parity generation circuit. Table 2-7 lists the BRS channel functions. The BRS channel does not have a busy or done flag and it cannot generate interrupt requests.

Table 2-7. BRS Channel Functions

| Function | Description |
|---|---|
| BRS:0 | Clear base register |
| BRS:14 | Write base register |
| BRS:17 | Write modes |

BRS:0 - Clear Base Register

This function clears the base register and the instruction stack parity error flag.

BRS:14 - Write Base Register

This function transfers accumulator bits $2^0$ through $2^6$ to the base register.

BRS:17 - Write Modes

This function writes the contents of accumulator bits $2^0$ and $2^1$ to two mode bits in the instruction stack. The mode bits affect the operation of the parity generation logic and are used only for maintenance purposes. When mode bit 0 is set, the parity bit that protects data bits $2^0$ through $2^7$ is inverted before it is stored. When mode bit 1 is set, the parity bit that protects data bits $2^8$ through $2^{15}$ is inverted before it is stored. If either mode bit is set when a parcel is written to the instruction stack, a parity error occurs when that parcel is read from the stack.

## External Channels

Channels 10 through 37 allow the IOPs to communicate with other components in the IOS and with devices outside the IOS. The channels are arranged in even/odd pairs, with the even channel usually used for input to an IOP and the odd channel usually used for output.

Figure 2-8 shows the IOP MUX and EIOP external channel configuration. The remainder of this section describes the features common to all external channels, and describes the specific purpose of each channel pair.

### Common Features

Although the external channels communicate with many different types of devices, much of the architecture is identical for all channels. The following subsections describe these common architectural features.

#### Registers

Each external channel has a pair of 16-bit registers that control data transfers to or from local memory. The local memory address register indicates the source or destination of the next data parcel. The parcel count register indicates the number of parcels remaining to be transferred. Each local memory address and parcel count register is protected with a single parity bit.

**Note:** Devices connected to some channels cannot transfer data to or from local memory and therefore do not use these registers. Specifically, HISP channels (IOP MUX channels 10 through 17) and some channel adapters (EIOP channels 30 through 37) cannot perform local memory transfers.

The programmer must load the local memory address and parcel count registers before beginning a local memory transfer. During the transfer, the local memory address register automatically increments and the parcel count register automatically decrements after each data parcel is transferred. The programmer can read either register at any time to determine how far the transfer has progressed.

**Note:** When an instruction reads the parcel count register, the inverted value (one's complement) of the parcel count is transmitted to the accumulator. To obtain the correct parcel count, invert the accumulator contents. This note does not apply to the parcel count register on the CCA1 channel adapter. The CCA1 parcel count register is a different register than the parcel count register described in the above paragraphs. Reading the CCA1 parcel count register transmits the correct (noninverted) parcel count to the accumulator.

#### Flags

Each external channel uses its busy and done flags to indicate the status of data transfers on the channel.

The busy flag indicates that a data transfer is in progress. For channels that can transfer data to or from IOP local memory, the busy flag sets when a local memory transfer is in

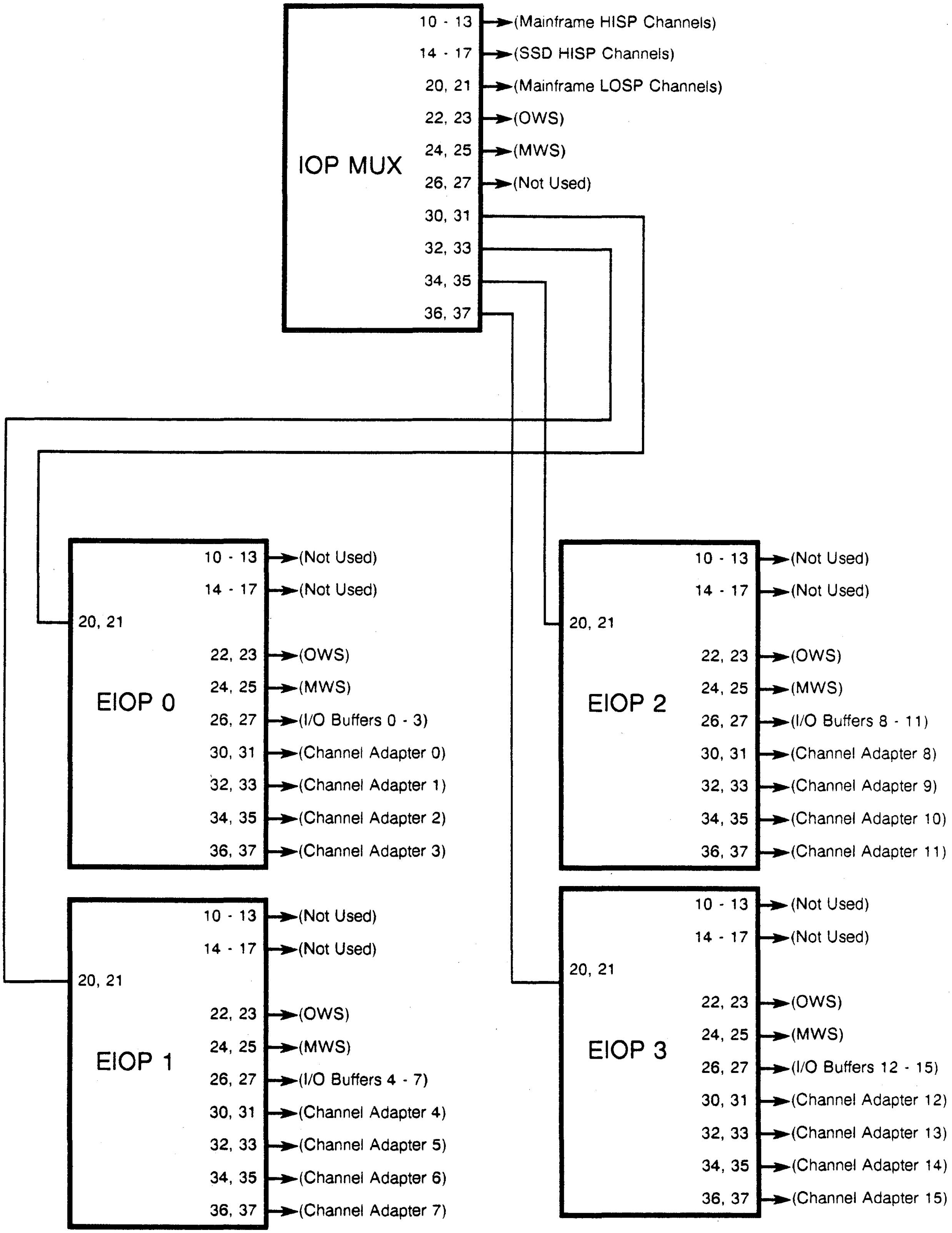

IOP MUX
10 - 13 (Mainframe HISP Channels)
14 - 17 (SSD HISP Channels)
20, 21 (Mainframe LOSP Channels)
22, 23 (OWS)
24, 25 (MWS)
26, 27 (Not Used)
30, 31
32, 33
34, 35
36, 37
EIOP 0
10 - 13 (Not Used)
14 - 17 (Not Used)
20, 21
22, 23 (OWS)
24, 25 (MWS)
26, 27 (I/O Buffers 0 - 3)
30, 31 (Channel Adapter 0)
32, 33 (Channel Adapter 1)
34, 35 (Channel Adapter 2)
36, 37 (Channel Adapter 3)
EIOP 1
10 - 13 (Not Used)
14 - 17 (Not Used)
20, 21
22, 23 (OWS)
24, 25 (MWS)
26, 27 (I/O Buffers 4 - 7)
30, 31 (Channel Adapter 4)
32, 33 (Channel Adapter 5)
34, 35 (Channel Adapter 6)
36, 37 (Channel Adapter 7)
EIOP 2
10 - 13 (Not Used)
14 - 17 (Not Used)
20, 21
22, 23 (OWS)
24, 25 (MWS)
26, 27 (I/O Buffers 8 - 11)
30, 31 (Channel Adapter 8)
32, 33 (Channel Adapter 9)
34, 35 (Channel Adapter 10)
36, 37 (Channel Adapter 11)
EIOP 3
10 - 13 (Not Used)
14 - 17 (Not Used)
20, 21
22, 23 (OWS)
24, 25 (MWS)
26, 27 (I/O Buffers 12 - 15)
30, 31 (Channel Adapter 12)
32, 33 (Channel Adapter 13)
34, 35 (Channel Adapter 14)
36, 37 (Channel Adapter 15)

Figure 2-8. IOP MUX and EIOP External Channel Configuration

progress. For channels that can transfer data to or from an I/O buffer, the busy flag sets when an I/O buffer transfer is in progress. For channels that can transfer data to both IOP local memory and an I/O buffer, the busy flag sets when either type of transfer is in progress.

The done flag indicates that a data transfer has completed. For channels that can transfer data to or from IOP local memory, the done flag sets when a local memory transfer completes. For channels that can transfer data to or from an I/O buffer, the done flag sets when an I/O buffer transfer completes. For channels that can transfer data to both IOP local memory and an I/O buffer, the done flag sets when either type of transfer completes.

Some channels set the busy and done flags simultaneously to indicate that an error occurred during a data transfer.

### Interrupts

Each external channel can generate an interrupt request at the end of a data transfer (when the channel done flag sets). An interrupt request occurs only if both the channel interrupt enable flag and the system interrupt enable flag are set. If either interrupt enable flag is clear, no interrupt request occurs.

Channel errors that set the busy and done flags simultaneously also generate interrupt requests, if both interrupt enable flags are set.

### Channel Functions

External channel functions are either processed on the IOP or sent to the external channel, depending on the channel function number. Channel functions 10, 11, 14, and 15 are processed on the IOP. Channel functions 0 through 7, 12, 13, 16, and 17 are sent to the external channel. The following information applies only to functions 0 through 7, 12, 13, 16, and 17.

When an external channel function issues, the IOP sends the following information to the channel adapter:

- Modifier bits M0 through M3 (*d* field or B register bits $2^5$ through $2^8$)
- *f* field bits $2^0$ through $2^3$ (channel function)
- Accumulator contents

Channel functions 12 and 13 return a status parcel to the IOP. The contents of the status parcel are determined by the external channel.

The IOP uses *d* field or B register bits $2^0$ through $2^4$ (the channel number) to determine which channel adapter should receive each channel function. The IOP does not send these bits to the channel adapter.

Because each channel adapter receives channel functions for at least two channel numbers, the modifier bits and accumulator distinguish functions intended for different channels. Each channel adapter uses either the modifier bits or the accumulator to determine the channel number. Each device interprets this information differently; there is no single channel number encoding scheme that applies to all devices.

### Channels 10 through 17

Channels 10 through 17 are used only by the IOP MUX; they serve no function in the EIOPs. These channels control data transfers between the I/O buffers and the HISP channels. Refer to Section 4 of this manual, "Mainframe and SSD Channels," for more information on IOP MUX channels 10 through 17.

### Channels 20 and 21

Channels 20 and 21 perform different functions in the IOP MUX than they do in the EIOPs. In the IOP MUX, these channels transfer data between IOP MUX local memory and the LOSP channels. Refer to Section 4, "Mainframe and SSD Channels," for more information on IOP MUX channels 20 and 21.

Channels 20 and 21 in the EIOPs transfer data between EIOP and IOP MUX local memory. Channel 20 in each EIOP receives data from the IOP MUX. Channel 21 transmits data to the IOP MUX. There is no data error correction or detection on these data transfers.

All transfers between an EIOP and the IOP MUX require that both IOPs initiate the transfer. For example, for a data transfer from an EIOP to the IOP MUX, the EIOP must start a transfer on its output channel (21) and the IOP MUX must start a transfer on its input channel (30, 32, 34, or 36, depending on which EIOP is sending the data). The transfer stops when the parcel count register in either IOP decrements to 0. Table 2-8 shows the channel functions for EIOP channel 20, input from IOP MUX (MXI). Table 2-9 shows the channel functions for EIOP channel 21, output to IOP MUX (MX0).

**Note:** The input channel can buffer 1 parcel of data even when it is not active. The buffered parcel can be either the first parcel of a transfer or a parcel received after the input channel's parcel count register has decremented to 0.

Table 2-8. Input from IOP MUX Channel Functions

| Function | Description |
|---|---|
| MXI:0 | Clear channel |
| MXI:1 | Start transfer |
| MXI:6 | Clear interrupt enable flag |
| MXI:7 | Set interrupt enable flag |
| MXI:10 | Read local memory address |
| MXI:11 | Read inverted parcel count |
| MXI:14 | Write local memory address |
| MXI:15 | Write parcel count |

MXI:0 - Clear Channel

This function aborts any data transfer in progress, clears the busy and done flags, and clears the 1-parcel input buffer.

MXI:1 - Start Transfer

This function sets the busy flag, clears the done flag, and starts a data transfer. Before the transfer can begin, the IOP MUX must start a transfer on its corresponding output channel. The transfer ends when either of these two conditions occurs:

- The channel's parcel count register decrements to 0 and 1 additional parcel is received. (If this condition occurs, the additional parcel can be recovered by reactivating the channel without clearing it.)

- The output channel's parcel count register decrements to 0.

MXI:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag.

MXI:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag.

MXI:10 - Read Local Memory Address

This function transfers the contents of the local memory address register to the accumulator.

MXI:11 - Read Inverted Parcel Count

This function transfers the inverted contents of the parcel count register to the accumulator.

**Note:** To obtain the correct parcel count, subtract the accumulator contents from $177777_8$.

MXI:14 - Write Local Memory Address

This function transfers the contents of the accumulator to the local memory address register.

Table 2-9. Output to IOP MUX Channel Functions

| Function | Description |
|---|---|
| MXO:0 | Clear channel |
| MXO:1 | Start transfer |
| MXO:6 | Clear interrupt enable flag |
| MXO:7 | Set interrupt enable flag |
| MXO:10 | Read local memory address |
| MXO:11 | Read inverted parcel count |
| MXO:14 | Write local memory address |
| MXO:15 | Write parcel count |

MXI:15 - Write Parcel Count

This function transfers the contents of the accumulator to the parcel count register.

MXO:0 - Clear Channel

This function aborts any data transfer in progress and clears the busy and done flags.

MXO:1 - Start Transfer

This function sets the busy flag, clears the done flag, and starts a data transfer. Before the transfer can begin, the IOP MUX must start a transfer on its corresponding input channel.

MXO:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag.

MXO:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag.

MXO:10 - Read Local Memory Address

This function transfers the contents of the local memory address register to the accumulator.

MXO:11 - Read Inverted Parcel Count

This function transfers the inverted contents of the parcel count register to the accumulator.

**Note:** To obtain the correct parcel count, subtract the accumulator contents from $177777_8$.

MXO:14 - Write Local Memory Address

This function transfers the contents of the accumulator to the local memory address register.

MXO:15 - Write Parcel Count

This function transfers the contents of the accumulator to the parcel count register.

### Channels 22 through 25

Channels 22 through 25 in each IOP transfer data between the IOP and the workstations. Data can be transferred across these channels under control of either the IOP or the workstation. To support workstation-controlled transfers, each of these channels has an additional local memory address and parcel count register that is controlled by the workstation. Refer to Section 6, "Cluster Interface/Workstation Interface," for more information on workstation data transfers.

### Channels 26 and 27

Channels 26 and 27 are used only by the EIOPs; they serve no function in the IOP MUX. These channels transfer data between local memory and the I/O buffer and partially control transfers between the I/O buffer and the channel adapters. Refer to Section 3, "I/O Buffer," for more information on these channels.

### Channels 30 through 37

Channels 30 through 37 perform different functions in the IOP MUX than they do in the EIOPs. In the EIOPs, these channels transfer data between local memory and the channel adapters, and they control transfers between the channel adapters and the I/O buffer. Refer to Section 5, "Channel Adapters," for more information on channel adapter data transfers.

Channels 30 through 37 in the IOP MUX transfer data between IOP MUX and EIOP local memory. Even channels 30 through 36 receive data from EIOP 0 through EIOP 3, respectively. Odd channels 31 through 37 transmit data to EIOP 0 through EIOP 3. There is no data error correction or detection on these data transfers.

All transfers between the IOP MUX and an EIOP require that both IOPs initiate the transfer. For example, to transfer data from the IOP MUX to EIOP 1, the IOP MUX must start a transfer on its output channel (33) and the EIOP must start a transfer on its input channel (20). The transfer stops when the parcel count register in either IOP decrements to 0.

Table 2-10 shows the channel functions for IOP MUX channels 30, 32, 34, and 36, input from EIOP (XIA). Table 2-11 shows the channel functions for IOP MUX channels 31, 33, 35, and 37, output to EIOP (XOA). The following subsections explain these channel functions.

Table 2-10. Input from EIOP Channel Functions

| Function | Description |
|---|---|
| XIA:0 | Clear channel |
| XIA:1 | Start transfer |
| XIA:6 | Clear interrupt enable flag |
| XIA:7 | Set interrupt enable flag |
| XIA:10 | Read local memory address |
| XIA:11 | Read inverted parcel count |
| XIA:14 | Write local memory address |
| XIA:15 | Write parcel count |

Table 2-11. Output to EIOP Channel Functions

| Function | Description |
|---|---|
| XOA:0 | Clear channel |
| XOA:1 | Start transfer |
| XOA:6 | Clear interrupt enable flag |
| XOA:7 | Set interrupt enable flag |
| XOA:10 | Read local memory address |
| XOA:11 | Read inverted parcel count |
| XOA:14 | Write local memory address |
| XOA:15 | Write parcel count |

**Note:** The input channel can buffer 1 parcel of data even when it is not active. The buffered parcel can be either the first parcel of a transfer or a parcel received after the input channel's parcel count register has decremented to 0.

### XIA:0 - Clear Channel

This function aborts any data transfer in progress, clears the busy and done flags, and clears the 1-parcel input buffer.

### XIA:1 - Start Transfer

This function sets the busy flag, clears the done flag, and starts a data transfer. Before the transfer can begin, the corresponding EIOP must start a transfer on its output channel. The transfer ends when either of these two conditions occurs:

- The channel's parcel count register decrements to 0 and 1 additional parcel is received. (If this condition occurs, the additional parcel can be recovered by reactivating the channel without clearing it.)
- The output channel's parcel count register decrements to 0.

### XIA:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag.

### XIA:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag.

### XIA:10 - Read Local Memory Address

This function transfers the contents of the local memory address register to the accumulator.

### XIA:11 - Read Inverted Parcel Count

This function transfers the inverted contents of the parcel count register to the accumulator.

**Note:** To obtain the correct parcel count, subtract the accumulator contents from $177777_8$.

### XIA:14 - Write Local Memory Address

This function transfers the contents of the accumulator to the local memory address register.

### XIA:15 - Write Parcel Count

This function transfers the contents of the accumulator to the parcel count register.

XOA:0 - Clear Channel

This function aborts any data transfer in progress and clears the busy and done flags.

XOA:1 - Start Transfer

This function sets the busy flag, clears the done flag, and starts a data transfer. Before the transfer can begin, the corresponding EIOP must start a transfer on its input channel.

XOA:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag.

XOA:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag.

XOA:10 - Read Local Memory Address

This function transfers the contents of the local memory address register to the accumulator.

XOA:11 - Read Inverted Parcel Count

This function transfers the inverted contents of the parcel count register to the accumulator.

**Note:** To obtain the correct parcel count, subtract the accumulator contents from $177777_8$.

XOA:14 - Write Local Memory Address

This function transfers the contents of the accumulator to the local memory address register.

XOA:15 - Write Parcel Count

This function transfers the contents of the accumulator to the parcel count register.

# CONTENTS

**FIGURES**

**FIGURES (continued)**

**TABLES**

# 3. I/O BUFFER

Each cluster in the I/O subsystem (IOS) contains 16 I/O buffers (numbered 0 through $17_8$) that multiplex to two high-speed (HISP) channel pairs. The I/O buffer performs the following operations:

- Transfers data to and from local memory of an auxiliary IOP (EIOP). These transfers are controlled by the I/O buffer input and output functions described later in this section.

- Transfers data to and from a CRI mainframe or SSD solid-state storage device (SSD) through HISP channels. These transfers are controlled by the low speed (LOSP) and HISP input and output functions provided in Section 4, "Mainframe and SSD Channels."

- Transfers data between a peripheral device and the I/O buffer through channel adapters. These transfers are controlled by the channel adapter input and output functions described in Section 5, "Channel Adapters."

Memory capacity for an I/O buffer is 64 Kwords (64-bit word plus 8 bits of single-error correction/double-error detection (SECDED) logic). The I/O buffer can transfer data at a rate of one 64-bit word per 18.75 nanoseconds (ns).

Each EIOP controls four I/O buffers (numbered 0 through 3). Channels 26 and 27 are assigned I/O buffer-to-local memory transfers, and local memory-to-I/O buffer transfers. Channel 26 is used when data is written to local memory from the I/O buffer. Channel 27 is used when data is read from local memory to the I/O buffer.

I/O buffer channels generate interrupt requests when the done flag sets at the completion of a data transfer. Refer to Section 2 of this manual, "I/O Processor," for more information on channel interrupts. Each channel also has a set of local memory address (LMA) and parcel count registers. Information on the LMA and parcel count registers is also provided in Section 2 of this manual.

Each I/O buffer has four I/O buffer pointers for flexibility in programming. A description of these pointers is provided in the following subsection.

## I/O BUFFER POINTERS

I/O buffer pointers specify locations in the I/O buffer to receive or transmit data. Each I/O buffer has four 16-bit pointers (A, B, a, and b). HISP channel and IOP local memory data transfers use pointers A and B. Channel adapter data transfers use pointers a and b. The I/O buffer pointers can be set to any beginning word address. These pointers are set at the beginning of a transfer and increment each time an address in the I/O buffer is referenced. It is the programmer's responsibility to ensure the correct pointer is initialized for each operation.

When writing to local memory from the I/O buffer, the I/O buffer pointer terminates at a buffer address two locations past the last word transferred to local memory. When reading from local memory to the I/O buffer, the I/O buffer pointer terminates at a buffer address one location past the last word transferred from local memory.

I/O buffer pointers can also be decremented for support of tape read backwards commands. Information about the tape read backwards commands will be provided in Section 5, "Channel Adapters," when available.

I/O buffer contention is resolved by allowing each pair of I/O buffer pointers access on alternating memory cycles. Refer to Figure 3-1 for an illustration of the I/O buffer memory cycles.

When the A and B pointers are in contention, a HISP transfer has priority over a local memory transfer. If simultaneous HISP or simultaneous local memory transfers to one are I/O buffer using the same pointer are attempted, the pointer may be incorrectly incremented.

When the a and b pointers are in contention, there is no priority scheme. A transfer in progress must be completed before the other transfer can begin.

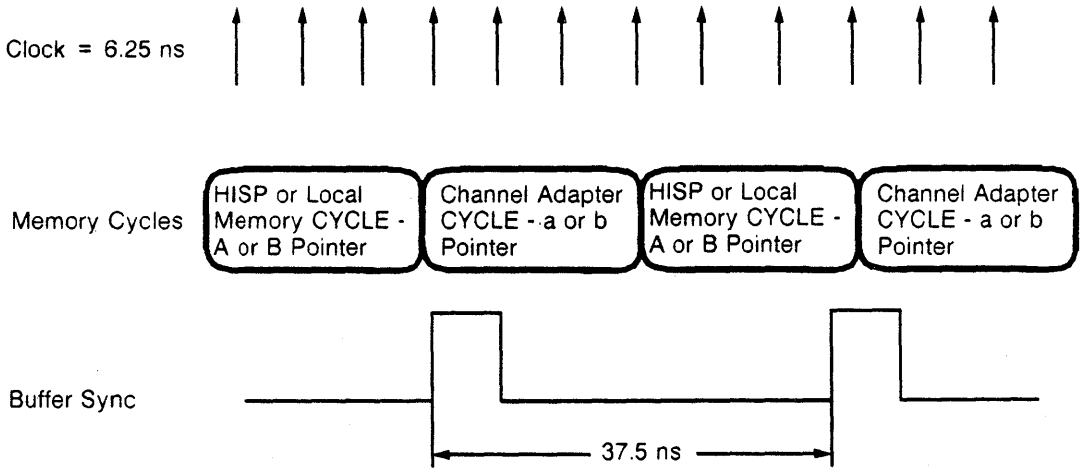
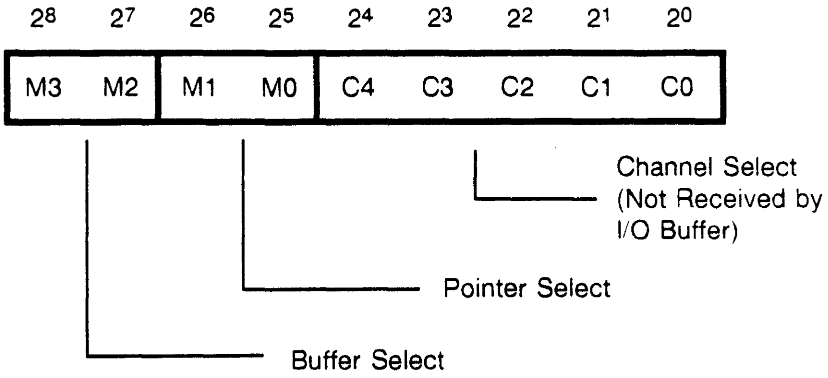

Clock = 6.25 ns
Memory Cycles
HISP or Local Memory CYCLE - A or B Pointer
Channel Adapter CYCLE - a or b Pointer
HISP or Local Memory CYCLE - A or B Pointer
Channel Adapter CYCLE - a or b Pointer
Buffer Sync
37.5 ns

Figure 3-1. I/O Buffer Memory Cycles

## CHANNEL FUNCTIONS

Channel functions provide control between the I/O buffer and IOP local memory. Channel functions also perform SECDED maintenance on the I/O data.

Certain I/O buffer channel functions (functions 1, 16, and 17) select the I/O buffer, I/O buffer pointer, and channel to use for a particular operation. These selections are made by setting or clearing the instruction *d* field or B register parameter bits for that function. Refer to Figure 3-2 for an illustration of the parameter bits and parameter bit selection options. The remainder of this section describes the I/O buffer input and output functions.



| M3 | M2 | Buffer | M1 | M0 | Pointer |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | a |
| 0 | 1 | 1 | 0 | 1 | b |
| 1 | 0 | 2 | 1 | 0 | A |
| 1 | 1 | 3 | 1 | 1 | B |

Figure 3-2. Parameter Bits and Parameter Bit Selections

## Input Functions

The I/O buffer input functions control data transfers from an I/O buffer to local memory of an IOP. Refer to Table 3-1 for a description of the I/O buffer input functions.

Table 3-1. BIA Channel Functions

| Function | Description |
|---|---|
| BIA:0 | Clear channel |
| BIA:1 | Start local memory transfer |
| BIA:6 | Clear interrupt enable flag |
| BIA:7 | Set interrupt enable flag |
| BIA:10 | Read local memory address |
| BIA:11 | Read inverted parcel count |
| BIA:12 | Read SECDED status |
| BIA:14 | Write local memory address |
| BIA:15 | Write parcel count |
| BIA:16 | Write upper buffer pointer address and diagnostic mode |
| BIA:17 | Write lower buffer pointer address |

### BIA:0 - Clear Channel

This function aborts any transfer in progress, clears the channel busy and done flags, and clears the SECDED status register (refer to function BIA:12). Accumulator bit $2^0$ must be a 0 to specify an I/O buffer-to-local memory transfer. This function does not use modifier bits M0 through M3.

### BIA:1 - Start Local Memory Transfer

This function sets the busy flag, clears the done flag, and starts a transfer. Accumulator bit $2^0$ must be a 0 to specify an I/O buffer to local memory transfer. This function uses modifier bits M0 through M3.

Data read from the I/O buffer is checked for validity with SECDED. If an error is detected, the error information is stored as a status word (refer to BIA:12). If either a single-bit error or a double-bit error occurs on the transfer, the transfer completes, but the busy flag stays set.

### BIA:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag. This function does not use modifier bits M0 through M3.

### BIA:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag. This function does not use modifier bits M0 through M3.

### BIA:10 - Read Local Memory Address

This function transfers the contents of the local memory address register to the accumulator. This function does not use modifier bits M0 through M3.

### BIA:11 - Read Inverted Parcel Count

This function transfers the inverted contents of the parcel count register to the accumulator. This function does not use modifier bits M0 through M3.

**Note:** To obtain the correct parcel count, subtract the accumulator contents from $177777_8$.

### BIA:12 - Read SECDED Status

This function transfers the SECDED status register contents to the accumulator and then clears the SECDED status register. Function BIA:0 also clears the SECDED status register. Function BIA:1 clears the SECDED status register at the beginning of each transfer. SECDED status from previous transfers is disregarded.

Refer to Figure 3-3 for an illustration of the BIA:12 status parcel format. Bits $2^6$ through $2^{13}$ contain the syndrome. The single-bit (bit $2^{15}$) error flag sets if syndrome bits are generated for a single-bit error. The double-bit (bit $2^{14}$) error flag sets if syndrome bits are generated for a double-bit error. Both error flags may simultaneously be set; however, the first single-bit syndrome to occur is written into the syndrome register (bits $2^6$ through $2^{13}$). Bits $2^0$ through $2^5$ are not used.

This function does not use modifier bits M0 through M3.

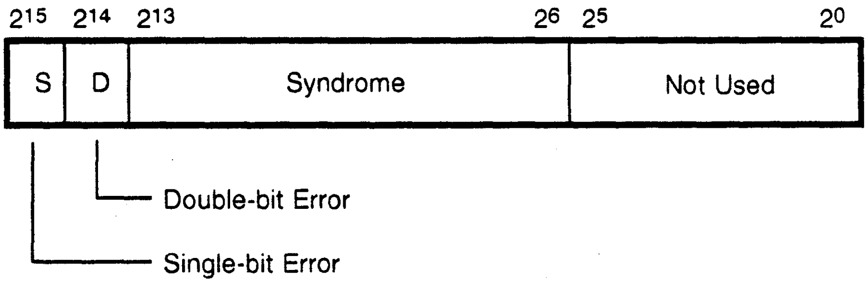


Figure 3-3. BIA:12 Status Parcel

### BIA:14 - Write Local Memory Address

This function transfers the contents of the accumulator to the local memory address register. This function does not use modifier bits M0 through M3.

**Note:** The local memory address should be sent to a multiple of four to correspond to a buffer word boundary.

### BIA:15 - Write Parcel Count

This function transfers the contents of the accumulator into the parcel count register. This function does not use modifier bits M0 through M3.

**Note:** The parcel count should be set to a multiple of four so that an integral number of buffer words are transferred.

### BIA:16 - Write Upper Buffer Pointer Address and Diagnostic Mode

This function transfers the contents of the accumulator to the upper buffer pointer address register and to the 3 diagnostic mode bits. The least significant accumulator bits are reserved for use as the upper bits of the pointer address. This function uses modifier bits M0 through M3.

Refer to Figure 3-4 for an illustration of the BIA:16 parameter bits. Accumulator bits $2^{0}$ through $2^{5}$ are transferred to bits $2^{16}$ through $2^{21}$ of the buffer pointer address register. (Bits $2^{0}$ through $2^{5}$ are currently not used.) Bits $2^{6}$ through $2^{8}$ modify the operation of the SECDED logic for maintenance purposes. These bits remain set until cleared by another BIA:16 function. Bits $2^{9}$ through $2^{15}$ are not used.

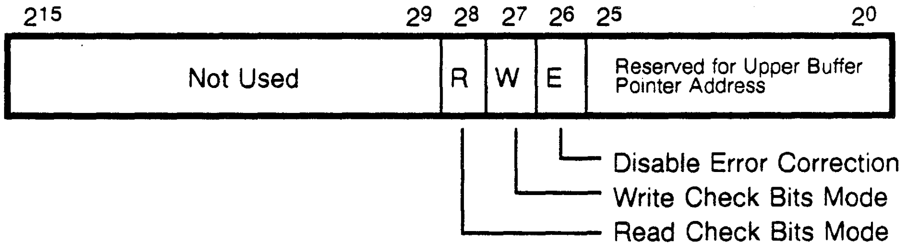


Figure 3-4. BIA:16 Parameter Bits

Bit $2^{6}$ sets the disable error correction function for data being read from the I/O buffers into local memory, or to the high-speed channels. All four I/O buffers of an IOP are controlled by this bit.

Bit $2^{7}$ sets the write check bits mode for local memory to I/O buffer transfers. In this mode, two local memory words are combined to form one I/O buffer word. Refer to Figure 3-5 for an illustration of the BIA:16 I/O buffer write check bits diagnostic mode. The data bits $2^{56}$ through $2^{63}$ of local memory word 0 are used as the check byte for the I/O buffer word 0. The data bits $2^{0}$ through $2^{63}$ of local memory word 1 are used as the I/O buffer word 0. The data bits $2^{56}$ through $2^{63}$ of local memory word 2 are used as the check byte for the I/O buffer word 1. The data bits $2^0$ through $2^{63}$ of local memory word 3 are used as the I/O buffer word 1. This pattern continues throughout the transfer.

Bit $2^8$ sets the read check bits mode. In this mode, data bits $2^{56}$ through $2^{63}$ of each word are replaced by the check byte of that word after error correction.

The write and read check bit modes diagnose the error detection and correction logic. The test word is written into the I/O buffer with a test set of check bits. When a word is read using the read check bits mode, the word and check bits are sent from the buffer to the SECDED logic. Corrected check bits are then placed in the upper bits of the data word so that they can be tested as a data field.

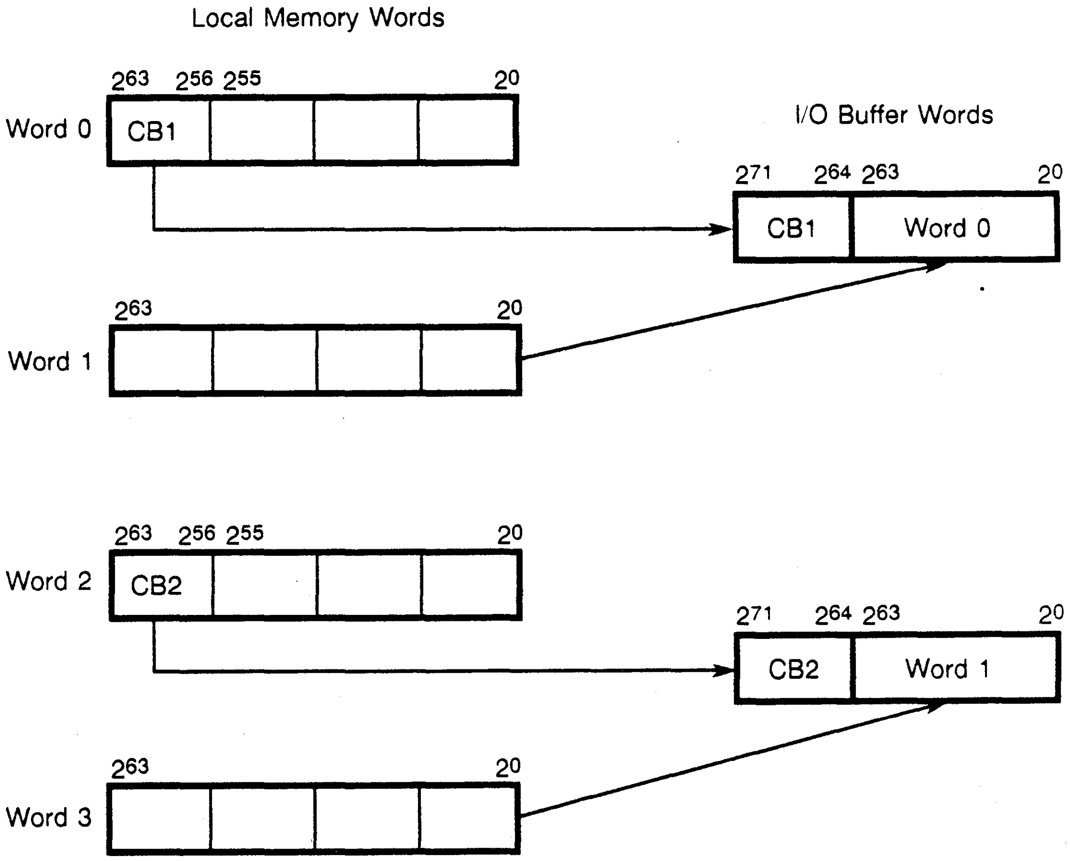


Figure 3-5. BIA:16 I/O Buffer Write Check Bits Diagnostic Mode

### BIA:17 - Write Lower Buffer Pointer Address

This function transfers the accumulator contents to the lower 16-bits of the buffer pointer register. (Because the I/O buffer currently contains 64 Kwords, each buffer pointer register is 16 bits long, and function BIA:17 loads the entire register.) This function uses B register or *d* field modifier bits M0 through M3 to select the pointer and the buffer.

## Output Functions

The I/O buffer output functions control data transfers from local memory of an IOP to an I/O buffer. Refer to Table 3-2 for a description of the I/O buffer output functions.

Table 3-2. BOA Channel Functions

| Function | Description |
|---|---|
| BOA:0 | Clear channel |
| BOA:1 | Start local memory transfer |
| BOA:6 | Clear interrupt enable flag |
| BOA:7 | Set interrupt enable flag |
| BOA:10 | Read local memory address |
| BOA:11 | Read inverted parcel count |
| BOA:14 | Write local memory address |
| BOA:15 | Write parcel count |
| BOA:16 | Write upper buffer pointer address and diagnostic mode |
| BOA:17 | Write lower buffer pointer address |

### BOA:0 - Clear Channel

This function aborts any transfer in progress and clears the channel busy and done flags. This function also clears the SECDED status register. Accumulator bit $2^0$ must be a 1 to specify a local memory-to-I/O buffer transfer. This function does not use modifier bits M0 through M3.

Data read from local memory is checked for validity with SECDED. If an error is detected, it is reported to local memory error channel 3.

### BOA:1 - Start Local Memory Transfer

This function sets the busy flag, clears the done flag, and starts a transfer. Accumulator bit $2^0$ must be a 1 to specify a local memory to I/O buffer transfer. This function uses modifier bits M0 through M3.

### BOA:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag. This function does not use modifier bits M0 through M3.

### BOA:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag. This function does not use modifier bits M0 through M3.

### BOA:10 - Read Local Memory Address

This function transfers the contents of the local memory address register to the accumulator. This function does not use modifier bits M0 through M3.

### BOA:11 - Read Inverted Parcel Count

This function transfers the inverted contents of the parcel count register to the accumulator. This function does not use modifier bits M0 through M3.

**Note:** To obtain the correct parcel count, subtract the accumulator contents from $177777_8$.

### BOA:14 - Write Local Memory Address

This function transfers the contents of the accumulator to the local memory address register. This function does not use modifier bits M0 through M3.

**Note:** The local memory address should be sent to a multiple of four to correspond to a buffer word boundary.

### BOA:15 - Write Parcel Count

This function transfers the contents of the accumulator to the parcel count register. This function does not use modifier bits M0 through M3.

**Note:** The parcel count should be set to a multiple of four so that an integral number of buffer words are transferred.

### BOA:16 - Write Upper Buffer Pointer Address and Diagnostic Mode

This function transfers the contents of the accumulator to the upper buffer pointer address register and to the 3 diagnostic mode bits. The least significant accumulator bits are reserved for use as the upper bits of the pointer address. This function uses modifier bits M0 through M3.

Refer to Figure 3-6 for an illustration of the BOA:16 parameter bits. Accumulator bits $2^0$ through $2^5$ are transferred to bits $2^{16}$ through $2^{21}$ of the buffer pointer address register. (Bits $2^0$ through $2^5$ are currently not used.) Bits $2^6$ through $2^8$ modify the operation of the SECDED logic for maintenance purposes. These bits remain set until cleared by another BIA:16 function. Bits $2^9$ through $2^{15}$ are not used.

Bit $2^6$ sets the disable error correction function for data being read from local memory into the I/O buffers. All four I/O buffers of an IOP are controlled by this bit.

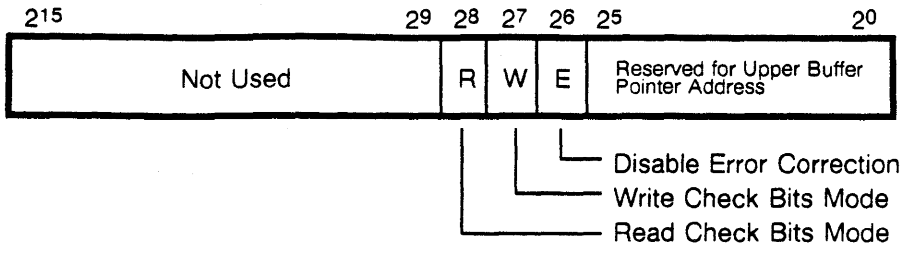


Figure 3-6. BOA:16 Parameter Bits

Bit $2^7$ sets the write check bits mode. In this mode, two local memory words are combined to form one I/O buffer word. Refer to Figure 3-7 for an illustration of the BOA:16 I/O buffer write check bits diagnostic mode. Data bits $2^{56}$ through $2^{63}$ of local memory word 0 are used as the check byte for I/O buffer word 0. Data bits $2^0$ through $2^{63}$ of local memory word 1 are used as I/O buffer word 0. Data bits $2^{56}$ through $2^{63}$ of local memory word 2 are used as the check byte for I/O buffer word 1. Data bits $2^0$ through $2^{63}$ of local memory word 3 are used as I/O buffer word 1. This pattern continues throughout the transfer.

Bit $2^8$ sets the read check bits mode for I/O buffer-to-local memory transfers. In this mode, data bits $2^{56}$ through $2^{63}$ of each word are replaced by the check byte of that word after error correction.

The write and read check bit modes diagnose the error detection and correction logic. The test word is written into the I/O buffer with a test set of check bits. When a word is read using the read check bits mode, the word and check bits are sent from the buffer to the SECDED logic. Corrected check bits are then placed in the upper bits of the data word so they can be tested as a data field.

### BOA:17 - Write Lower Buffer Pointer Address

This function transfers the accumulator contents to the lower 16-bits of the buffer pointer register. (Because the I/O buffer currently contains 64 Kwords, each buffer pointer register is 16 bits long, and function BOA:17 loads the entire register.) This function uses B register or *d* field modifier bits M0 through M3 to select the pointer and the buffer.

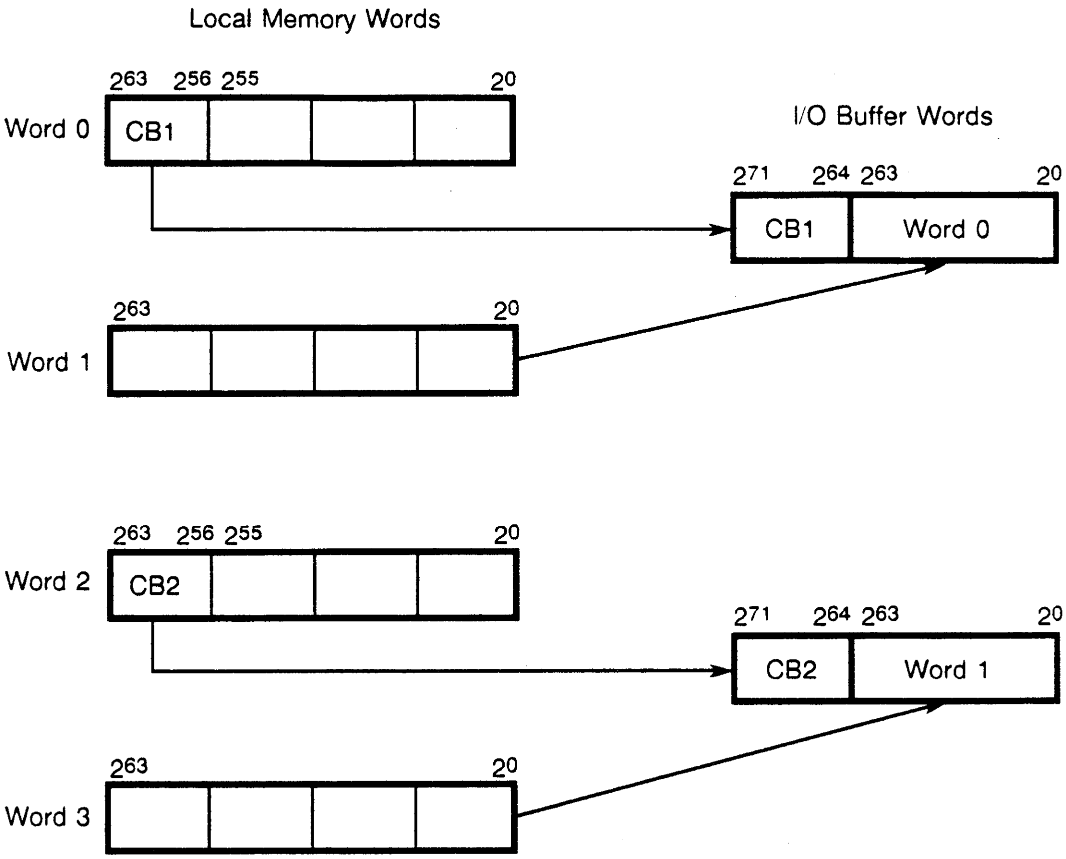


Figure 3-7. BOA:16 I/O Buffer Write Check Bits Diagnostic Mode

# CONTENTS

## FIGURES

**FIGURES (continued)**

**TABLES**

# 4. MAINFRAME AND SSD CHANNELS

Each cluster of the I/O subsystem (IOS) contains two types of I/O channels that provide communication with the mainframe and the SSD solid-state storage device (SSD). Low-speed (LOSP) channels operate at 6 or 50 Mbytes/s and transmit control information between the IOS and the mainframe. High-speed (HISP) channels operate at 100 or 200 Mbytes/s and transmit data between the IOS and mainframe and between the IOS and SSD.

The LOSP and HISP channels are organized in pairs, each having one input channel and one output channel. Each cluster contains one LOSP channel pair and two HISP channel pairs. The LOSP channel pair transmits control information between local memory of the multiplexer I/O processor (IOP MUX) and central memory of the mainframe. One HISP channel pair transmits data between the I/O buffers and mainframe central memory. The other pair transmits data between the I/O buffers and the SSD.

The following subsections describe the LOSP and HISP channel pairs in detail.

## LOW-SPEED CHANNEL PAIR

The LOSP channel pair transmits data between local memory of the IOP MUX and central memory of the mainframe; both channels can operate simultaneously. Each channel is 16 bits (1 parcel) wide and contains 4 parity bits for data error detection. Each channel can operate at either 6 Mbytes/s or 50 Mbytes/s. (CRAY Y-MP computer systems currently operate at 6 Mbytes/s; 50-Mbyte/s operation is reserved for future use.) The IOP MUX and the mainframe share control of the LOSP channel pair.

The following subsections describe the operation of LOSP channels. First, registers used by the LOSP channels are described. Then the 6-Mbyte/s and 50-Mbyte/s protocols are explained. Next, channel errors are described. Finally, channel programming (using the IOP MUX) is explained.

### Low-speed Channel Registers

Each LOSP channel has two registers that control data transfers across the channel. The local memory address register selects a location in IOP MUX local memory to receive or transmit the channel data. The parcel count register indicates the number of parcels remaining to be transferred. Both registers are programmed through the IOP MUX.

The programmer must initialize the local memory address and parcel count registers before starting a data transfer. The local memory address register increments and the parcel count register decrements after each parcel is transmitted or received. When the parcel count register decrements to 0, the transfer is complete.

**Note:** The LOSP input channel actually has two parcel count registers, one located in the IOP MUX and one located in the LOSP channel interface. The programmer must make sure both parcel count registers contain the same value before starting a data transfer.

## Low-speed Channel Flags

Each LOSP channel has two flags that indicate the status of the channel. The busy flag sets at the beginning of each data transfer and clears at the completion of the transfer. The done flag sets at the completion of each data transfer and remains set until it is cleared by a program.

If an error is detected during a LOSP data transfer, the busy and done flags set simultaneously. Both flags remain set until cleared by software.

## Low-speed Channel Signals

Each LOSP channel (input or output) contains 16 data bits, 4 parity bits, and 3 control signals (Ready, Resume, and Disconnect). Five additional control signals (Master Clear, Deaddump, Deadstart, MCU Interrupt, and Device Master Clear) correspond to each channel pair. Another control signal, the Programmed Interrupt, allows the IOS to selectively interrupt mainframe CPUs. Figure 4-1 shows all the signals.

The following subsections describe the operation of the LOSP channel signals during channel operations. First, 6-Mbyte/s data transfers are discussed, followed by 50-Mbyte/s transfers, auxiliary operations, and finally Programmed Interrupts.

### 6-Mbyte/s Data Transfers

Data transfers using the 6-Mbyte/s protocol can contain anywhere from 1 to 65,536 parcels. The protocol is almost identical for input and output transfers. The normal input sequence is as follows:

1. The mainframe places a data parcel and 4 parity bits on the channel and activates the Ready signal to indicate that data is available on the channel.

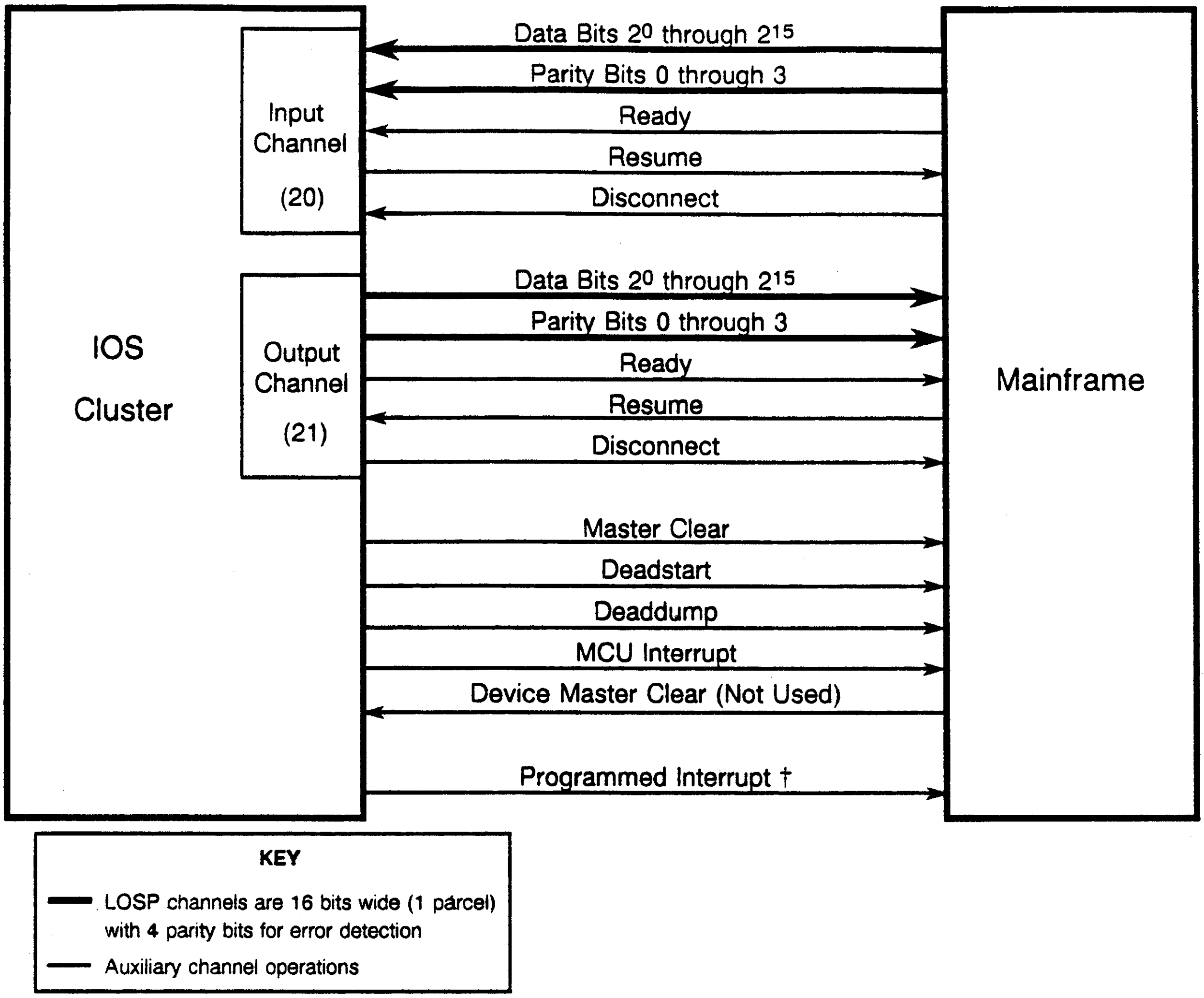
2. The IOS reads the data and parity bits from the channel and checks for parity errors. It activates the Resume signal to indicate that it is ready to receive another data parcel.

3. The IOS writes the data to IOP MUX local memory, then increments the local memory address register and decrements the parcel count register.

4. Steps 1 through 3 are repeated until all data is transferred across the channel.

5. The data transfer stops when either of the following conditions occurs, which generates an interrupt request:

   - The mainframe activates the Disconnect signal to indicate that the transfer is complete.



† The Programmed Interrupt signal is not included in all computer systems. To verify if your computer system is equipped with this feature, refer to Table 4-1 in this section.

Figure 4-1. LOSP Channel Signals

- The parcel count register decrements to 0. If this condition occurs, the IOS stops sending the Resume signal, which prevents the mainframe from sending more data.

If the mainframe performs step 1 when the IOS input channel is not active, the IOS does not perform step 2. Instead, the IOS latches the Ready signal in the ready held flag. When the IOP MUX activates the input channel, the ready held flag indicates that a parcel of data is already on the channel and that the IOS can immediately perform step 2.

The normal 6-Mbyte/s output sequence is as follows:

1. The IOS reads 1 parcel of data from IOP MUX local memory and places the parcel and 4 parity bits on the channel. It activates the Ready signal to indicate that data is available on the channel.

2. The IOS increments the local memory address register and decrements the parcel count register.

3. The mainframe reads the data and parity bits from the channel and checks for parity errors. It activates the Resume signal to indicate that it is ready to receive more data.

4. Steps 1 through 3 are repeated until all data is transferred across the channel.

5. The data transfer stops when either of the following conditions occurs:

   - The parcel count register decrements to 0. This condition activates the Disconnect signal to indicate the transfer is complete and generates an interrupt request.

   - The mainframe stops sending the Resume signal. This condition prevents the IOS from sending more data; it does not generate an interrupt request.

### 50-Mbyte/s Data Transfers

Data transfers using the 50-Mbyte/s protocol can contain between 4 and 65,536 parcels but must be a multiple of 4 parcels. The protocol is similar to the 6-Mbyte/s protocol, except that the receiving device sends the Resume signal only once for each 4 parcels received. The normal 50-Mbyte/s input sequence is as follows:

1. The mainframe places a data parcel and 4 parity bits on the channel. It activates the Ready signal to indicate that data is available on the channel.

2. The IOS reads the data and parity bits from the channel into a 4-parcel buffer and checks for parity errors.

3. Steps 1 and 2 are repeated three more times until the IOS receives a total of 4 parcels.

4. The IOS transfers the data from the buffer to IOP MUX local memory. The local memory address register increments and the parcel count register decrements after each parcel is written.

5. The IOS activates the Resume signal to indicate that it is ready to receive 4 more data parcels. This step does not necessarily occur after step 4; it can occur anytime after the mainframe has sent the first parcel in the 4-parcel group.

6. Steps 1 through 5 are repeated until all data is transferred across the channel.

7. The transfer stops when any one of the following conditions occurs:

   - The mainframe activates the Disconnect signal to indicate that the transfer is complete. If this condition occurs, the IOS acknowledges the Disconnect signal by sending a final Resume signal. This condition generates an interrupt request.

- The IOP MUX's parcel count register decrements to 0. If this condition occurs, the IOS stops sending the Resume signal, preventing the mainframe from sending more data. This condition generates an interrupt request.

- The IOP MUX sends a Disconnecting Resume signal. The Disconnecting Resume signal is a longer-than-normal Resume signal and is under IOP program control. This condition does not generate an interrupt request.

If the mainframe sends data to the IOS when the IOS input channel is not active, the IOS reads the data into the 4-parcel buffer but does not transfer the data to IOP MUX local memory. Instead, the IOS latches the Ready signal in a flag called the ready held flag. When the IOP MUX activates the input channel, the ready held flag indicates that 1 or more parcels of data are in the 4-parcel buffer and that the data can immediately be transferred to IOP MUX local memory.

The normal 50-Mbyte/s output sequence is as follows:

1. The IOS reads 4 data parcels from IOP MUX local memory. The local memory address register increments and the parcel count register decrements after each parcel is read.

2. The IOS places a data parcel and 4 parity bits on the channel. It activates the Ready signal to indicate that data is available on the channel.

3. The mainframe reads the data and parity bits from the channel and checks for parity errors.

4. Steps 2 and 3 are repeated three more times until the IOS transmits all 4 parcels.

5. The mainframe activates the Resume signal to indicate that it is ready to receive 4 more data parcels. This step does not necessarily occur after step 4; it can occur anytime after the mainframe has sent the first parcel in the 4-parcel group.

6. Steps 1 through 5 are repeated until all data is transferred across the channel.

7. The transfer stops when any one of the following conditions occurs:

   - The parcel count register decrements to 0. This condition activates the Disconnect signal to indicate that the transfer is complete. The mainframe acknowledges the Disconnect signal by sending a final Resume signal. This condition generates an interrupt request.

   - The mainframe stops sending the Resume signal. This condition prevents the IOS from sending more data; it does not generate an interrupt request.

   - The mainframe sends a Disconnecting Resume signal. The Disconnecting Resume signal is a longer-than-normal Resume signal and is under mainframe program control. This condition generates an interrupt request.

### Auxiliary Channel Operations

Five control signals in the LOSP channel pair perform auxiliary operations and are not used during data transfers. Four of the signals (Master Clear, Deaddump, Deadstart, and MCU Interrupt) are programmable through the IOP MUX and allow the IOP MUX to control operation of the mainframe. The fifth signal (Device Master Clear) allows the mainframe to control the operation of the cluster. The following list describes each of these signals.

Signal

Description

Master Clear

Enables the IOP MUX to force a master clear in the mainframe. When this signal sets, all program activity in the mainframe stops. When the signal clears, the mainframe performs an exchange sequence and begins issuing instructions.

Deaddump

Forces the mainframe to transfer data to the IOS using the standard 6-Mbyte/s or 50-Mbyte/s protocol. The data transfer begins on the 0-to-1 transition of the Deaddump signal; the 1-to-0 transition has no effect on data transfers. The data transfer begins at mainframe central memory address 0 and continues as long as the IOS accepts the data. When the parcel count register decrements to 0, the IOS stops sending the Resume signal, which stops the transfer.

Deadstart

Causes an I/O master clear in the mainframe and forces the mainframe to receive data from the IOS using the standard 6-Mbyte/s or 50-Mbyte/s protocol. The I/O master clear occurs on the 0-to-1 transition of the Deadstart signal; the data transfer begins on the 1-to-0 transition. The data transfer begins at mainframe central memory address 0 and continues as long as the IOS sends data. When the parcel count register decrements to 0, the IOS sends the Disconnect signal, which stops the transfer.

MCU Interrupt

Enables the IOP MUX to force an interrupt in the mainframe. The interrupt occurs on the 0-to-1 transition of this signal; the 1-to-0 transition has no effect.

Device Master Clear

Performs no operation in the IOS.

### Programmed Interrupt

This feature controls the Transmit Programmed Interrupt signals that are sent from the IOS to the mainframe. The Transmit Programmed Interrupt signal, issued through an IOP MUX, enables the IOS to selectively interrupt mainframe CPUs. A successive signal, issued through the same IOP MUX, clears the Transmit Programmed Interrupt.

The Transmit Programmed Interrupt signal enables any I/O cluster from the IOS-E to interrupt any CPU within the mainframe configuration. The MCUI within the mainframe receives a 16-bit Transmit Programmed Interrupt signal from the IOS-E. The MCUI interrogates the signal and identifies which CPU to interrupt. Each bit of the

Transmit Programmed Interrupt signal corresponds to a CPU within the mainframe. For example, setting bit $2^3$ would interrupt CPU 3 in the mainframe. Figure 4-2 shows the Transmit Programmed Interrupt signal path from the IOS-E to the mainframe.

A CRAY Y-MP system configuration generally contains the same number of CPUs as IOS clusters. A CRAY Y-MP mainframe is expandable to a maximum configuration of 16 CPUs. The maximum configuration of I/O clusters within one IOS-E chassis is eight; therefore, a system configuration containing a mainframe with 16 CPUs requires two IOS-E chassis, each containing eight I/O clusters. A system containing two IOS-E chassis will send two Transmit Programmed Interrupt signals to the MCUI in the mainframe. The MCUI will logically OR these two signals together to generate one 16-bit signal identifying which CPU is interrupted.

Refer to Table 4-1 for the list of computers systems that have the Transmit Programmed Interrupt signal capability.

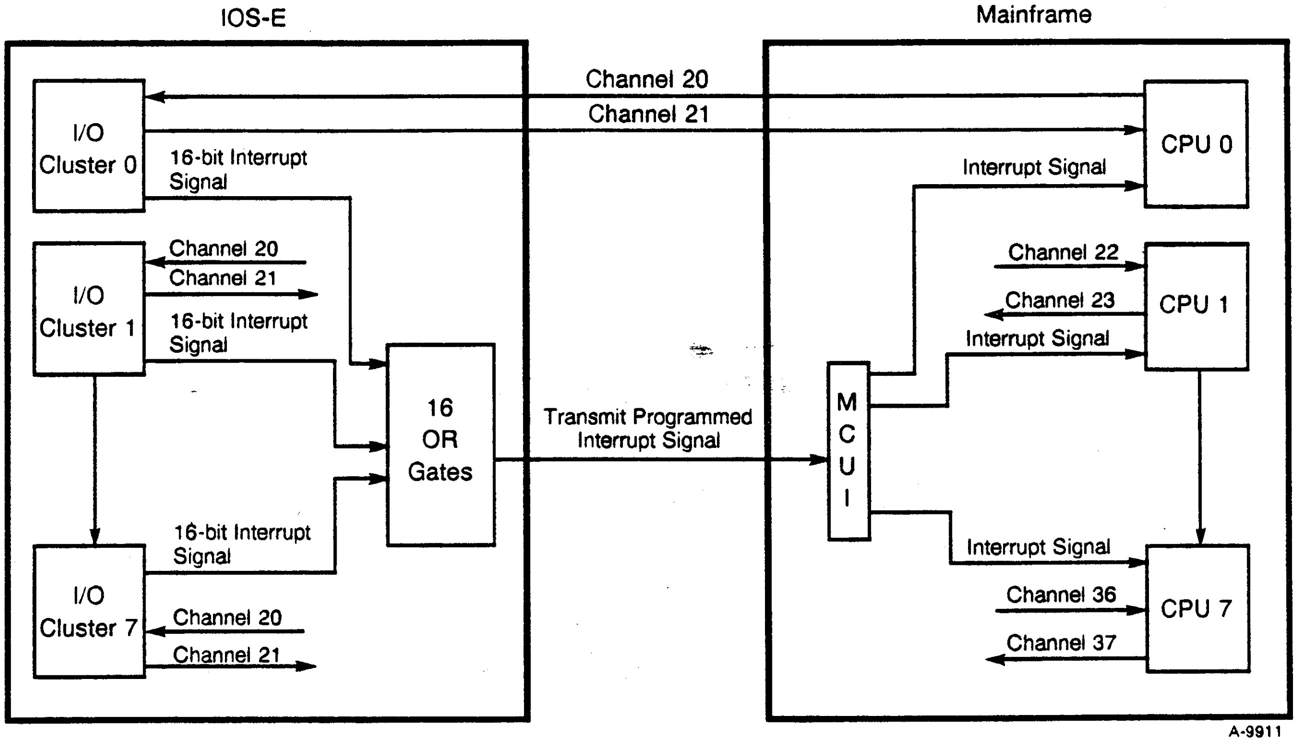


Figure 4-2. Transmit Programmed Interrupt Signal Paths

Table 4-1. Computer Systems with Programmed Interrupts

| Computer System | Serial Number |
|---|---|
| CRAY Y-MP2E | S/N 1637 and up |
| CRAY Y-MP4E | S/N 1912 and up |
| CRAY Y-MP8I | S/N 1709 and up |
| CRAY Y-MP8E | S/N 1802 and up |

## Low-speed Channel Errors

Two types of errors can occur on the LOSP channels: parity errors and control sequence errors. Parity errors occur when data is corrupted during a transfer. Control sequence errors occur when the Ready, Resume, and Disconnect signals are sent in an improper order.

The following subsections discuss parity errors and control sequence errors. Because the errors that can occur are identical regardless of channel direction (mainframe to IOS or IOS to mainframe), the terms *receiving device* and *transmitting device* are used instead of *IOS* and *mainframe*.

### Parity Errors

Parity errors that occur during a LOSP channel data transfer are detected by the receiving device; they are not communicated back to the transmitting device. When a parity error occurs, a parity error flag sets in the receiving device and the data transfer continues. The receiving device zeros all data received after the parity error. The busy flag stays set after the transfer is complete.

### Control Sequence Errors

Control sequence errors can occur when a channel is transferring data or when a channel is idle. Some of the errors can occur when the channel is in either 6-Mbyte/s or 50-Mbyte/s mode, while other errors can occur only when the channel is in 50-Mbyte/s mode.

The following subsections describe all the control sequence errors. The input channel of the receiving device detects some of these errors; the output channel of the transmitting device detects other errors. After a sequence error occurs, the busy flag and done flags stay set.

Input Channel Errors

LOSP input channels detect four types of errors: unexpected ready received errors, disconnect-with-ready-held errors, disconnect-on-odd-parcel-boundary errors, and parity errors.

An unexpected ready received error occurs when the receiving device receives one more Ready signal than expected. The receiving device expects to receive one Ready signal (6 Mbytes/s) or four Ready signals (50 Mbytes/s) after each Resume signal. If the receiving device receives two Ready signals (6 Mbytes/s) or five Ready signals (50 Mbytes/s), the unexpected ready received flag sets, but the transfer continues to run.

A disconnect-received-with-ready-held error occurs if the receiving device receives the Disconnect signal at an unexpected time. When the receiving device receives a Ready signal, it sets its ready held flag until it can send the Resume signal. If the receiving device receives the Disconnect signal before it has sent the Resume signal, the disconnect received with ready held flag sets, and the transfer is aborted.

A disconnect-received-on-odd-parcel-boundary error can occur only during 50-Mbyte/s data transfers. The receiving device expects to receive parcels of data in multiples of 4. If the receiving device receives the Disconnect signal when the number of parcels received is not a multiple of 4, the disconnect-received-on-odd-parcel-boundary flag sets and the transfer is aborted.

### Output Channel Errors

LOSP output channels detect four errors: unexpected resume-received error, extra-resume-received error, parcel-count error, and disconnecting-resume error.

An unexpected-resume-received error occurs if the transmitting device receives a Resume signal when the channel is not active. The unexpected resume flag sets, but no other action occurs.

An extra resume-received error can occur only during 50-Mbyte/s data transfers. When operating at 50 Mbytes/s, the transmitting device expects to receive a Resume signal only once for each group of four Ready signals. If two Ready signals are received, the extra resume flag sets, and the transfer is aborted.

A parcel count error can only occur during 50-Mbyte/s operations. It occurs at the end of a data transfer if the contents of the parcel count register is not 0. A parcel count error indicates that the contents of the parcel count register was not initially a multiple of four or that the parcel count register decremented incorrectly.

A Disconnecting Resume error can occur only during 50-Mbyte/s operations, when the receiving device receives the Disconnecting Resume signal. (The Disconnecting Resume signal is a longer-than-normal Resume signal.) The disconnecting resume error sets the disconnecting resume flag and aborts the transfer.

## Low-speed Channel Functions

The IOP MUX uses I/O channels 20 and 21 to program the LOSP channels. Channel 20 controls the LOSP input channel, and channel 21 controls the output channel. The following subsections describe the channel functions that the IOP MUX uses to control these channels.

## Input Channel Functions

This subsection explains the functions that control the LOSP input channel, IOP MUX channel 20. Table 4-2 lists these functions.

Table 4-2. LOSP Input Channel Functions

| Function | Description |
|---|---|
| MIA:0 | Clear channel |
| MIA:1 | Start transfer |
| MIA:2 | Clear ready held flag |
| MIA:5 | Set channel speed |
| MIA:6 | Clear interrupt enable flag |
| MIA:7 | Set interrupt enable flag |
| MIA:10 | Read local memory address |
| MIA:11 | Read inverted parcel count |
| MIA:12 | Read status |
| MIA:13 | Read error codes |
| MIA:14 | Write local memory address |
| MIA:15 | Write parcel count |
| MIA:17 | Disable internal busy flags |

### MIA:0 - Clear Channel

This function aborts any data transfer in progress and clears the busy and done flags.

### MIA:1 - Start Transfer

This function transfers the accumulator contents to the parcel count register on the LOSP channel interface, sets the busy flag, clears the done flag, and starts a data transfer.

**Note:** When this function issues, the accumulator must contain the same value as the parcel count register in the IOP MUX. The easiest way to ensure that the accumulator contains the correct value is to load the IOP MUX parcel count register (function MIA:15) immediately before starting the transfer (function MIA:1).

MIA:2 - Clear Ready Held Flag

This function performs two different operations, depending on whether the channel is programmed for 6-Mbyte/s or 50-Mbyte/s operation. In 6-Mbyte/s mode, this function should be used only when the channel is inactive. It clears the ready held flag and causes the IOP MUX to disregard a mainframe-initiated data transfer.

In 50-Mbyte/s mode, this function may be used whether the channel is active or inactive. The function clears the ready held flag and sends a Disconnecting Resume signal, stopping any data transfer that the mainframe has requested or that is in progress.

MIA:5 - Set Channel Speed

This function uses accumulator bit $2^0$ to set the operating speed of both the input and the output channel. If bit $2^0$ is 0, the channels operate at 6 Mbytes/s. If bit $2^0$ is 1, the channels operate at 50 Mbytes/s. The channels retain the selected operating speed until another MIA:5 function issues.

Accumulator bits $2^1$ through $2^{15}$ are not used.

**Note:** The LOSP channels do not power up in a known operating speed. Therefore function MIA:5 must be used after every power up to select the operating speed. Master clear commands (system, cluster, and processor master clear) do not affect the channel operating speed.

MIA:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag.

MIA:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag.

MIA:10 - Read Local Memory Address

This function transfers the contents of the local memory address register to the accumulator.

MIA:11 - Read Inverted Parcel Count

This function transfers the inverted contents of the parcel count register to the accumulator.

**Note:** To obtain the correct parcel count, subtract the accumulator contents from $177777_8$.

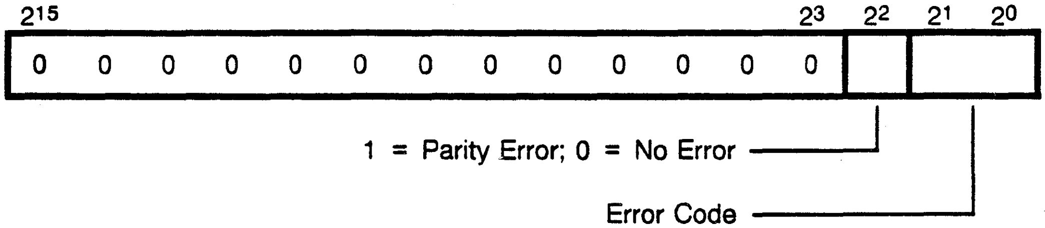
### MIA:12 - Read Status

This function transfers the contents of the ready held flag to bit $2^0$ of the accumulator. Accumulator bits $2^1$ through $2^{15}$ are 0.

### MIA:13 - Read Error Codes

This function transfers the contents of the input channel error flags to the accumulator, then clears the error flags. (Refer to the "Low-speed Channel Errors" subsection in this section for more information on the error flags.)

Figure 4-3 shows the format of the error flags in the accumulator. Bits $2^0$ and $2^1$ contain the encoded control sequence error flags. Bit $2^2$ contains the parity error flag.



| $2^1$ | $2^0$ | Error |
|---|---|---|
| 0 | 0 | No Error |
| 0 | 1 | Unexpected Ready Received |
| 1 | 0 | Disconnect Received on Odd Parcel Boundary † |
| 1 | 1 | Disconnect Received with Ready Held |

† 50-Mbyte/s mode only.

Figure 4-3. MIA:13 Status Parcel

### MIA:14 - Write Local Memory Address

This function transfers the contents of the accumulator to the local memory address register.

### MIA:15 - Write Parcel Count

This function transfers the contents of the accumulator to the parcel count register in the IOP MUX.

**Note:** This function does not load the parcel count register on the IOP channel interface. Function MIA:1 loads this register when it starts a data transfer.

MIA:17 - Disable Internal Busy Flag

This function uses accumulator bit $2^8$ to enable (bit $2^8=0$) or disable (bit $2^8=1$) the channel 20 busy flag that is internal to the IOP. This flag must be enabled in the EIOPs and disabled in the IOP MUX. Accumulator bits $2^0$ through $2^7$ and $2^9$ through $2^{15}$ are not used.

**Note:** Following any master clear command (system, cluster, or processor master clear), the channel 20 internal busy flag is enabled, which is the correct state for the EIOPs. To allow proper LOSP channel operation, the IOP MUX must execute an MIA:17 function (with accumulator bit $2^8$ equal to 1) after each master clear command.

### Output Channel Functions

This subsection explains the functions that control the LOSP output channel, IOP MUX channel 21. Table 4-3 lists these functions.

Table 4-3. LOSP Output Channel Functions

| Function | Description |
|---|---|
| MOA:0 | Clear channel |
| MOA:1 | Start transfer |
| MOA:3 | Clear or set inhibit disconnect mode |
| MOA:4 | Send external functions |
| MOA:5 | Send disconnect signal |
| MOA:6 | Clear interrupt enable flag |
| MOA:7 | Set interrupt enable flag |
| MOA:10 | Read local memory address |
| MOA:11 | Read inverted parcel count |
| MOA:12 | Read status |
| MOA:13 | Read error codes |
| MOA:14 | Write local memory address |
| MOA:15 | Write parcel count |
| MOA:16 | Transmit programmed interrupt signal |
| MOA:17 | Disable internal busy flag |

MOA:0 - Clear Channel

This function aborts any data transfer in progress and clears the busy and done flags. Modifier bit M0 must be set.

MIA:1 - Start Transfer

This function sets the busy flag, clears the done flag, and starts a data transfer. Modifier bit M0 must be set.

MOA:3 - Clear or Set Inhibit Disconnect Mode

This function uses accumulator bit $2^0$ to clear or set the inhibit disconnect mode. Modifier bit M0 must be set. If the mode is clear (bit $2^0 = 0$), the channel sends a Disconnect signal at the end of each data transfer. If the mode is set (bit $2^0 = 1$), the channel does not send a Disconnect signal. The selected operating mode is retained until another MOA:3 function issues. Accumulator bits $2^1$ through $2^{15}$ are not used.

Function MOA:3 is useful for transferring long data blocks. If the inhibit disconnect mode is set, the IOS can perform successive data transfers without having to reactivate the mainframe input channel after each transfer; the mainframe treats the successive transfers as one large transfer. Before the final transfer, the IOP MUX should execute function MOA:3 with accumulator bit $2^0$ equal to 0; this function causes the IOS to automatically send a disconnect signal at the end of the transfer.

**Note:** The LOSP output channel does not power up in a known mode. Therefore, function MOA:3 must be used after every power-up sequence to select the mode. Master clear commands do not affect the mode.

MOA:4 - Send External Functions

This function controls the auxiliary signals that are sent from the IOS to the mainframe. Modifier bit M0 must be set. (Refer to the "Auxiliary Channel Operations" subsection in this section for more information on these signals.)

Accumulator bits $2^0$ through $2^3$ set or clear the auxiliary signals; the signals remain set or clear until another MOA:4 function issues. Figure 4-4 shows the signal controlled by each accumulator bit.

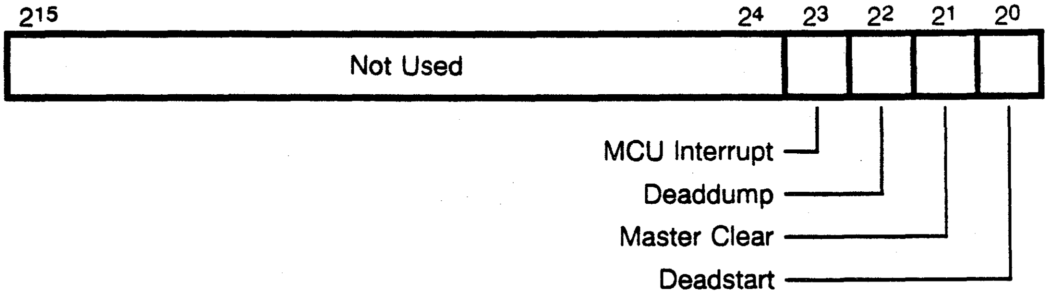


Figure 4-4. MOA:4 Command Parcel

### MOA:5 - Send Disconnect Signal

This function sends a Disconnect signal to the mainframe. Modifier bit M0 must be set. The Disconnect signal sets for a predetermined interval, then clears. This function is used primarily for channel error recovery.

### MOA:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag. Modifier bit M0 must be set.

### MOA:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag. Modifier bit M0 must be set.

### MOA:10 - Read Local Memory Address

This function transfers the contents of the local memory address register to the accumulator. Modifier bit M0 must be set.

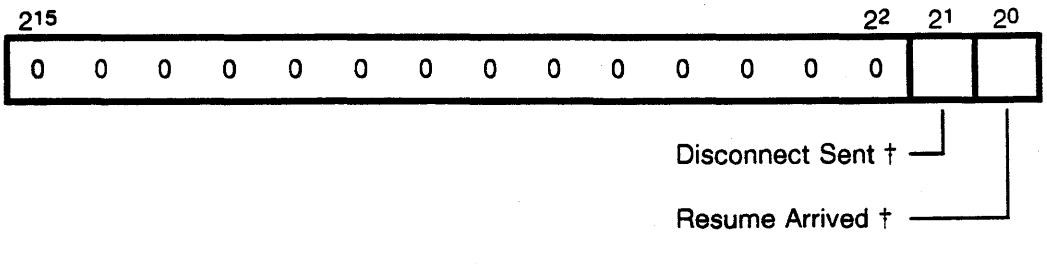
### MOA:11 - Read Inverted Parcel Count

This function transfers the inverted contents of the parcel count register to the accumulator. Modifier bit M0 must be set.

**Note:** To obtain the correct parcel count, subtract the accumulator contents from $177777_8$.

### MOA:12 - Read Status

This function transfers the contents of two flags to bits $2^0$ and $2^1$ of the accumulator. Modifier bit M0 must be set. Figure 4-5 shows the format of the flags in the accumulator. Use this function only in 50-Mbyte/s mode.



† 50-Mbyte/s mode only.

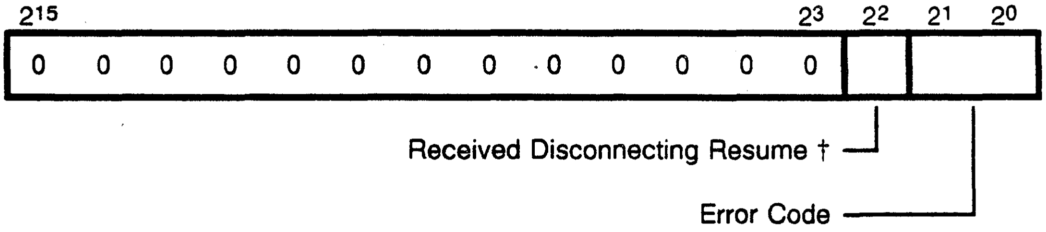
Figure 4-5. MOA:12 Status Parcel

The disconnect sent flag sets when the channel transmits the Disconnect signal at the end of the data transfer. The flag clears when the mainframe responds to the Disconnect signal by transmitting a Resume signal. This flag indicates that the mainframe has received and stored all the data.

The resume arrived flag clears each time the channel sends the first parcel (and corresponding Ready signal) in a 4-parcel group. The flag sets when the mainframe responds with a Resume signal. This flag is used primarily to help analyze errors.

## MOA:13 - Read Error Codes

This function transfers the contents of the output channel error flags to the accumulator, then clears the error flags. Modifier bit M0 must be set. (Refer to the "Low-speed Channel Errors" subsection in this section for more information on the error flags.)
Figure 4-6 shows the format of the error codes in the accumulator. Bits $2^0$ and $2^1$ contain encoded control sequence error flags. Bit $2^2$ contains the received disconnecting resume flag.



| $\underline{2^1}$ | $\underline{2^0}$ | Error |
|---|---|---|
| 0 | 0 | No Error |
| 0 | 1 | Parcel Count Error † |
| 1 | 0 | Unexpected Resume Received |
| 1 | 1 | Extra Resume Received |

† 50-Mbyte/s mode only.

Figure 4-6. MOA:13 Status Parcel

## MOA:14 - Write Local Memory Address

This function transfers the contents of the accumulator to the local memory address register. Modifier bit M0 must be set.

## MOA:15 - Write Parcel Count

This function transfers the contents of the accumulator to the parcel count register. Modifier bit M0 must be set.

## MOA:16 - Transmit Programmed Interrupt Signals

This function controls the Programmed Interrupt signals that are sent from the IOS-E to the mainframe. Each I/O cluster generates a 16-bit Programmed Interrupt signal that is sent to the PINT within an IOS-E. The PINT then logically ORs corresponding bits from each 16-bit I/O Cluster Interrupt signal. After completing this logical operation, one

16-bit Transmit Programmed Interrupt signal is created and sent to the MCUI within the mainframe. Each bit within one Transmit Programmed Interrupt signal corresponds to a CPU in the mainframe. For example, setting bits $2^0$, $2^3$, $2^6$ of the Transmit Programmed Interrupt signal will cause an interrupt to occur in CPUs 0, 3, and 6 in the mainframe. Refer to Figure 4-2. A CRAY Y-MP mainframe can be expanded to accomodate 16 CPUs.

All IOP MUXs in the same chassis share Programmed Interrupt signals. Any IOP MUX in the chassis can set any Programmed Interrupt signal; the signal remains set until cleared by the same IOP MUX. If two IOP MUXs in the same chassis attempt to interrupt the same CPU in quick succession and the first interrupt has not cleared, the second interrupt is not recognized. Therefore, the Transmit Programmed Interrupt signal remains set until it is cleared by another function MOA:16 or an IOP Master Clear is issued by the same IOP MUX. Modifier bit M0 must be set.

### MOA:17 - Disable Internal Busy Flag

This function is identical to function MIA:17, and is generally not used. Modifier bit M0 must be set. Refer to "MIA:17 - Disable Internal Busy Flag," earlier in this section for more information on these functions.

## HIGH-SPEED CHANNEL PAIRS

Each cluster contains two HISP channel pairs. Each pair consists of an input and an output channel, both of which may be active simultaneously. One HISP channel pair transmits data between the IOS and the mainframe; the second pair transmits data between the IOS and SSD. Each channel is 64 bits wide and contains 8 single-error correction/double-error detection (SECDED) bits. Each HISP channel can operate at either 100 Mbytes/s or 200 Mbytes/s (software selectable), depending on the capabilities of the mainframe and SSD.

High-speed data transfers to the mainframe can contain anywhere from 1 to 65,535 words. The HISP channels transmit data in 16-word blocks. All data blocks, except the final block of a transfer, contain exactly 16 words. The final block can contain from 1 to 16 words.

**Note:** Because of restrictions in the SSD, all data transfers to the SSD are limited to multiples of 64-word blocks.

The I/O buffers receive all HISP channel input data and transmit all HISP channel output data. Each of the 16 I/O buffers in the cluster can receive data from or transmit data to any HISP channel; however, each buffer can transmit data to or receive data from only one HISP channel at a time. The buffers use pointer A or pointer B for all high-speed transfers. (Refer to Section 3, "I/O Buffer," for more information on the I/O buffers and pointers.)

The IOS initiates and controls all high-speed data transfers. It determines the number of words to be transferred, selects an I/O buffer, and selects the starting addresses in both the I/O buffer and the external device (mainframe or SSD). Under normal operating conditions, operation of the HISP channels is transparent to the external devices; an external device cannot monitor or control HISP channel operation. However, if certain error conditions occur, an external device can abort a transfer.

The IOP MUX and an EIOP work together to initiate a HISP channel data transfer. The IOP MUX determines the number of words to be transferred, determines the starting address in the external device, selects an I/O buffer, and starts the transfer. When the transfer completes, the IOP MUX receives an interrupt request. The EIOP that controls the selected buffer determines the starting address in the buffer.

The following subsections describe the operation of the HISP channels. The first subsection describes the division of each channel into two subchannels. Subsequent subsections describe registers, channel signals, channel errors, and channel functions.

## High-speed Subchannels

Within the IOS, the control logic (not the data paths) for each HISP channel is divided into two subchannels. Each subchannel contains a complete set of control and status registers for the channel. The subchannels allow successive data transfers to proceed without a delay for the channel to be reprogrammed between transfers. That is, while a transfer using one subchannel is proceeding, the other subchannel can be programmed with the parameters of the next transfer. When the first transfer completes, the second transfer begins immediately.

The HISP channels are programmed through IOP MUX channels 10 through 17. Each channel number corresponds to one HISP subchannel. Even channel numbers are input channels, and odd channel numbers are output channels. Channels 10 through 13 are the mainframe channels, and channels 14 through 17 are SSD channels. Figure 4-7 shows the configuration of all the channels.

**Note:** Do not confuse subchannel A and subchannel B with buffer pointers A and B. Either subchannel can transfer data using either pointer.

## High-speed Channel Registers

Each HISP subchannel has two control registers. The starting address register indicates the starting memory address in the mainframe or SSD. The transfer word count register indicates the number of words to be transferred.

The programmer must load the starting address before starting a data transfer. Loading the word count register initiates the transfer. When the transfer begins, the contents of the starting address and transfer word count registers are transmitted to the mainframe or SSD. A copy of the transfer word count register is also kept in the IOS. After each data word is transferred across the channel, the starting address register increments and the transfer word count register (both copies) decrements. When the transfer word count register decrements to 0, the transfer is complete.

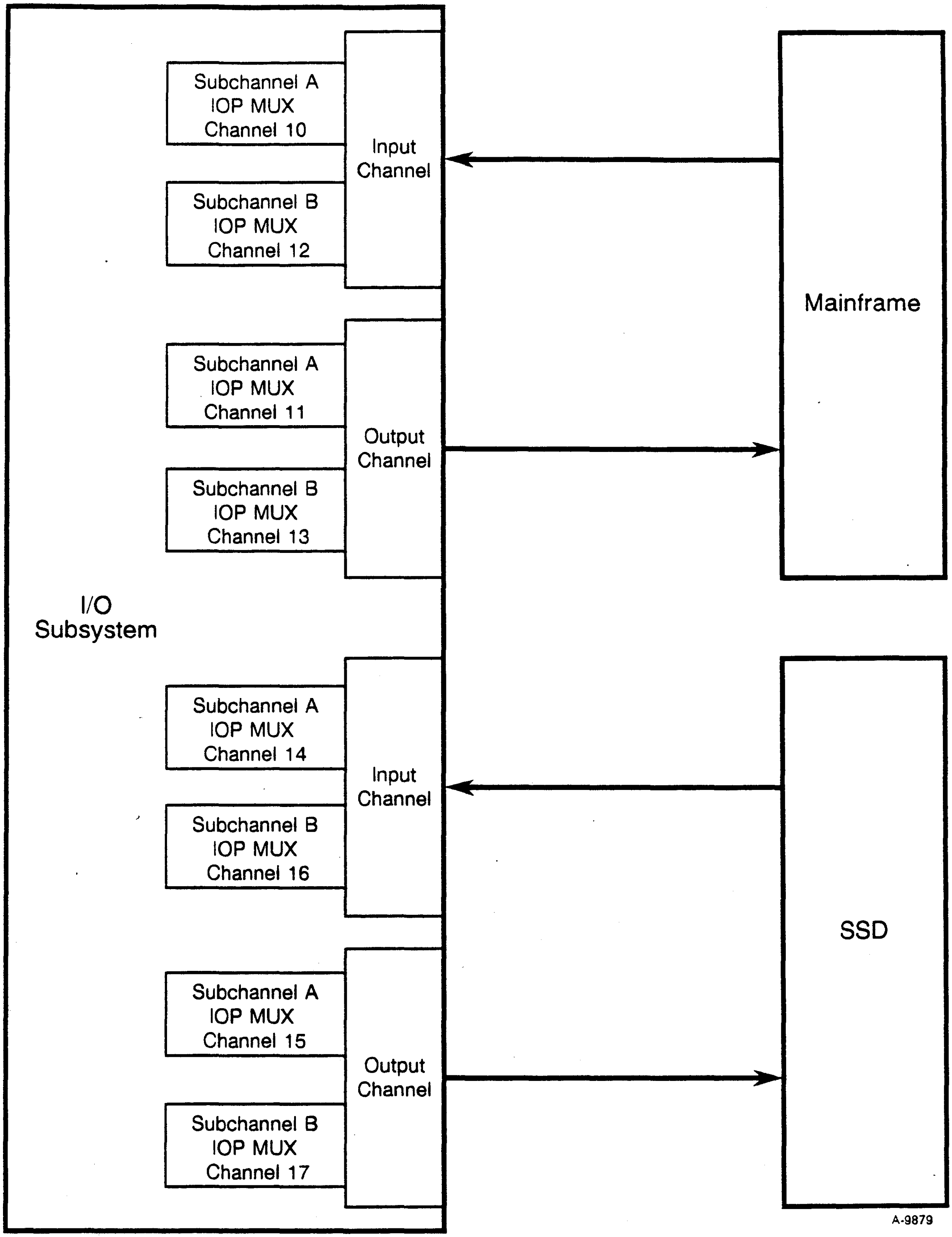


Figure 4-7. HISP Channel Configuration

## High-speed Channel Signals

The signals used by each HISP channel (input or output) can be divided into two categories:

- Address signals:
  - 1 Clear Channel signal
  - 1 Transmit Address signal
  - 1 Address Ready signal
  - 16 address channel bits (numbered 0 through 15)
  - 4 address parity bits (numbered 0 through 3)
  - 1 Address Error signal

- Data transfer signals:
  - 1 Transmit Data signal
  - 1 Data Ready signal
  - 64 data bits (numbered $2^0$ to $2^{63}$)
  - 8 data check bits (numbered 0 through 7)
  - 1 last word flag
  - 1 Uncorrectable Error signal
  - 1 Disable Error-correction signal (input channels only)

Figure 4-8 shows the address and data transfer signals for a HISP input channel; Figure 4-9 shows the signals for an output channel.

The address signals prepare the mainframe or SSD for a data transfer. They clear the channel and transmit the contents of the starting address and transfer word count registers to the mainframe and SSD. These signals can be programmed to operate at 100 or 200 Mbytes/s and are identical for input and output channels.

The data signals perform the actual data transfer. They can be programmed to operate at 100 or 200 Mbytes/s, independent of the operating speed of the address signals.

The following subsections explain the signals in detail.

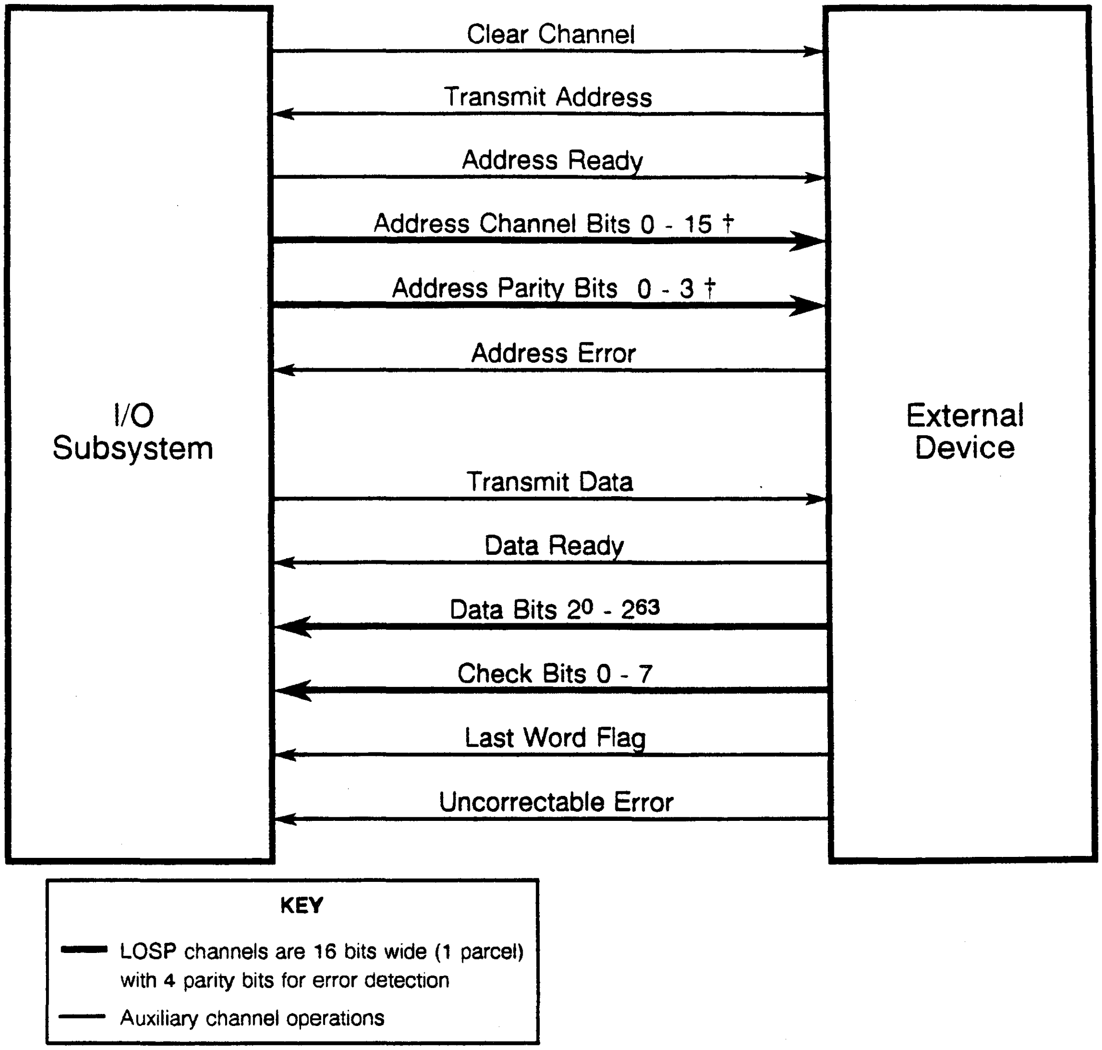
### Address Signals

The IOS prepares the external device for a data transfer by sending a Clear Channel signal. This signal stops any transfer already in progress on the channel and prepares the external device to receive address information. The external device then sends a Transmit Address signal to indicate that it is ready to receive address channel information.

The address channel transmits the contents of the starting address and transfer word count registers. In 100-Mbyte/s mode, the starting address and transfer word count registers are transmitted in three 16-bit parcels using address channel bits 0 through 15.

In 200-Mbyte/s mode, the registers are transmitted in five 12-bit words using address channel bits 0 through 11; address channel bits 12 through 15 are not used.

Address parity bits 0 through 3 protect the address channel information. Each parity bit protects 4 address channel bits. Parity bit 0 protects bits 0 through 3, parity bit 1 protects



† Address channel bits 12 through 15 and address parity bit 3 are not used in 200-Mbyte/s mode.

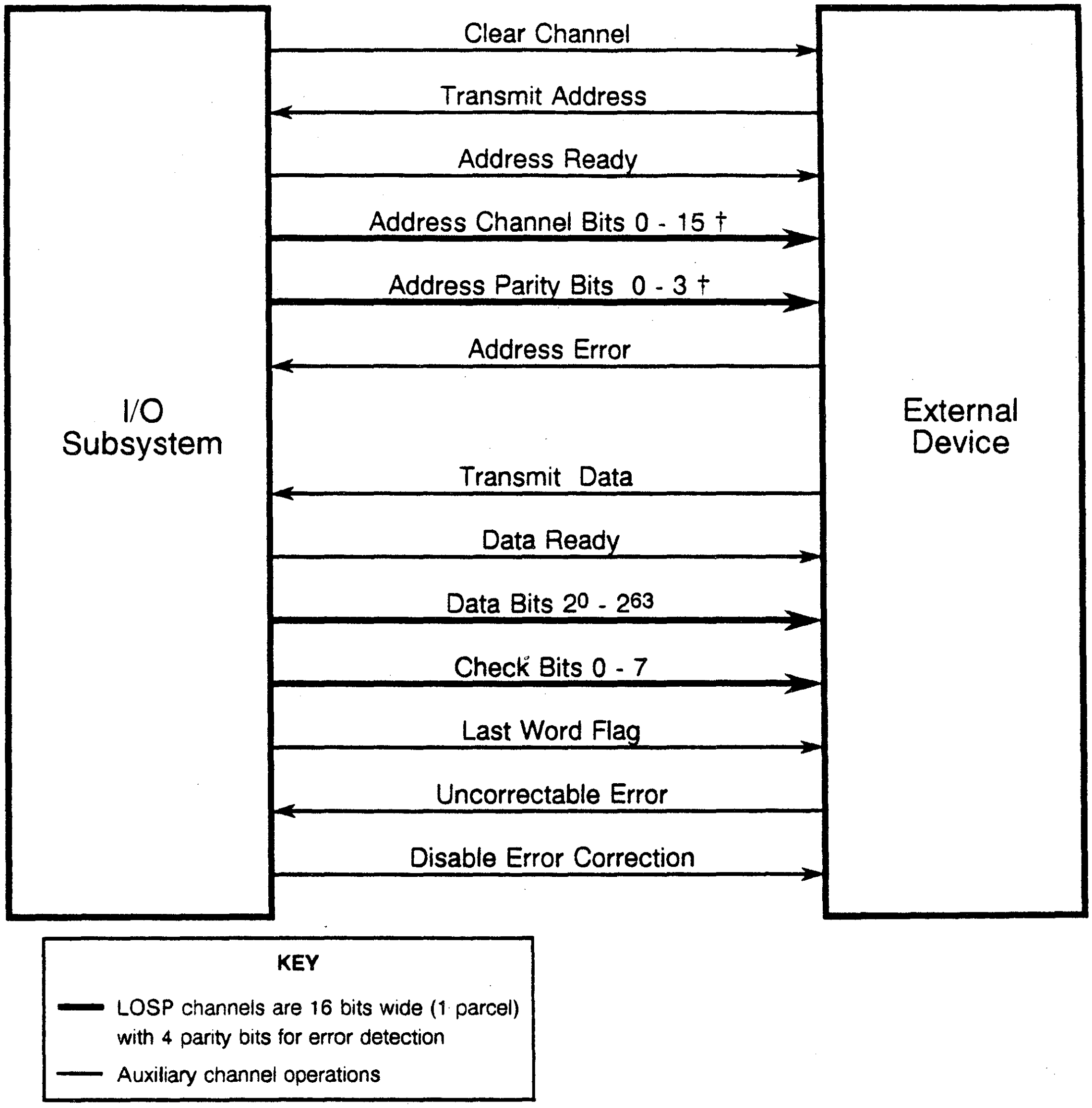
Figure 4-8. HISP Input Channel Signals

bits 4 through 7, and so on. In 100-Mybte/s mode, all 4 parity bits are used. In 200-Mbyte/s mode, only parity bits 0 through 2 are used.

Each time the IOS places a parcel of information (and 4 parity bits) on the address channel, it activates the Address Ready signal to indicate that the information is available. The IOS then deactivates the Address Ready signal until the next parcel is ready. The external device activates the Address Error signal if it detects an error during the transmission of address information. Address errors are discussed in detail in the "High-speed Channel Errors" subsection in this section.

### Data Transfer Signals

A data transfer begins after all 3 parcels (100-Mbyte/s mode) or 5 parcels (200-Mbyte/s mode) have been transmitted across the address channel. The signal sequence is as follows:



† Address channel bits 12 through 15 and address parity bit 3 are not used in 200-Mbyte/s mode.

A-9881

Figure 4-9. HISP Output Channel Signals

1. The receiving device activates the Transmit Data signal to indicate that it is ready to receive data.

2. The transmitting device places a 64-bit data word and 8 check bits on the channel and activates the Data Ready signal to indicate that the data is available.

3. Steps 1 and 2 are repeated 15 more times until the entire 16-word data block is transmitted. (The final block of the transfer may contain fewer than 16 words.)

4. When the receiving device is ready to receive another data block, it activates the Transmit Data signal.

5. Steps 1 through 4 are repeated until all data is transferred across the channel.

6. The transmitting device activates the Last Word Flag signal to indicate that the transfer is complete.

The external device activates the Uncorrectable Error signal if it detects an error during the transmission of address information or data. Errors are discussed in detail in the next subsection.

For channel maintenance, the IOS can send a Disable Error-correction signal to the external device. This signal affects both the input and the output channel. When the signal is set, SECDED errors occurring on the input channel are not corrected; the data and check bits are written to memory without modification. SECDED errors that occur between memory and the output channel are not corrected; the data and check bits are transferred across the channel without modification.

## High-speed Channel Errors

Several types of errors can occur on the HISP channels. The IOS detects some of the errors, while the mainframe or SSD detect other errors. All errors detected by the mainframe or SSD, except correctable data errors, are reported to the IOS. The following paragraphs discuss all the types of errors.

Address errors occur when the address channel is active and are detected by the mainframe or SSD. There are four types of address errors:

- Unexpected Address Ready signal received error. This error occurs if the mainframe or SSD receives four (100-Mbyte/s mode) or six (200-Mbyte/s mode) Address Ready signals in succession or if it receives an Address Ready signal during data transmission.

- Unexpected Transmit Data signal received error (IOS input channels only). This error occurs if the mainframe receives a Transmit Data signal when it is expecting to receive an Address Ready signal.

- Unexpected Data Ready signal received error (IOS output channels only). This error occurs if the mainframe or SSD receives a Data Ready signal when it is expecting to receive an Address Ready signal.

- Address parity errors.

If an address error occurs, the mainframe or SSD activates the Address Error and Uncorrectable Error signals and prevents the data transfer from beginning.

An active error occurs if the IOS receives a Transmit Address signal during a data transfer. An active error aborts the data transfer.

A word count error occurs if the transfer word count registers maintained by the IOS and the mainframe or SSD do not decrement to 0 simultaneously. The receiving device detects word count errors; if the mainframe or SSD is the receiving device and detects a word count error, it sets the Uncorrectable Error signal to notify the IOS. A word count error aborts the data transfer.

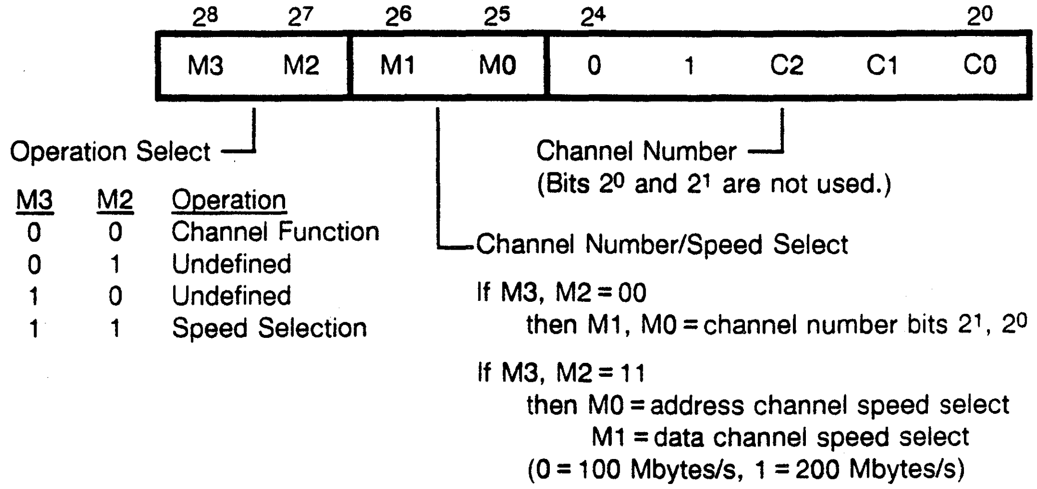
Correctable and uncorrectable data errors can occur in either the IOS or in the mainframe or SSD. Correctable errors do not abort the data transfer. If the mainframe or SSD detects a correctable error, it does not report the error to the IOS.

Uncorrectable data errors abort the data transfer. If the mainframe or SSD detects an uncorrectable data error, it sets the Uncorrectable Error signal to notify the IOS.

## High-speed Channel Instructions

The IOP MUX uses I/O channels 10 through 13 to control the mainframe HISP channels and I/O channels 14 through 17 to control the SSD HISP channels.

Figure 4-10 shows how the IOP MUX uses the *d* instruction field or B register in HISP channel instructions. (Refer to "External Channels" in Section 2 of this manual for general information on channel instructions.) Modifier bits M2 and M3 (*d* field or B register bits $2^7$ and $2^8$) determine the operation of the instruction. If bits M2 and M3 are both 0, the instruction performs a normal channel function. If bits M2 and M3 are both 1, the instruction selects an operating speed for the mainframe or SSD channels.



| $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | | | | $2^0$ |
|---|---|---|---|---|---|---|---|---|
| M3 | M2 | M1 | M0 | 0 | 1 | C2 | C1 | C0 |

Operation Select

| M3 | M2 | Operation |
|---|---|---|
| 0 | 0 | Channel Function |
| 0 | 1 | Undefined |
| 1 | 0 | Undefined |
| 1 | 1 | Speed Selection |

Channel Number
(Bits $2^0$ and $2^1$ are not used.)

Channel Number/Speed Select

If M3, M2 = 00
then M1, M0 = channel number bits $2^1$, $2^0$

If M3, M2 = 11
then M0 = address channel speed select
M1 = data channel speed select
(0 = 100 Mbytes/s, 1 = 200 Mbytes/s)

Figure 4-10. HISP Channel Instruction *d* Field or B Register

Table 4-4 shows all combinations of *d* field or B register bits that apply to the HISP channels. (Bit $2^3$ must be 1 and bit $2^4$ must be 0; other values apply to devices other than the HISP channels.)

The following subsections describe how HISP channel instructions control the HISP channels. First, the speed selection functions are described. Then the input and output channel functions are described.

Table 4-4. HISP Channel Instruction Decode

| *d* Field or B Register | | | | | | | | | | Operation |
|---|---|---|---|---|---|---|---|---|---|---|
| Binary | | | | | | | | | Octal | |
| M3 $2^8$ | M2 $2^7$ | M1 $2^6$ | M0 $2^5$ | C4 $2^4$ | C3 $2^3$ | C2 $2^2$ | C1 $2^1$ | C0 $2^0$ | | |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | X | X | 010 - 013 | Channel 10 (input from logical subchannel A) function |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | X | X | 050 - 053 | Channel 11 (output to logical subchannel A) function |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | X | X | 110 - 113 | Channel 12 (input from logical subchannel B) function |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | X | X | 150 - 153 | Channel 13 (output to logical subchannel B) function |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | X | X | 014 - 017 | Channel 14 (input from logical subchannel A) function |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | X | X | 054 - 057 | Channel 15 (output to logical subchannel A) function |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | X | X | 114 - 117 | Channel 16 (input from logical subchannel B) function |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | X | X | 154 - 157 | Channel 17 (output to logical subchannel B) function |
| 0 | 1 | X | X | 0 | 1 | X | X | X | | Undefined |
| 1 | 0 | X | X | 0 | 1 | X | X | X | | Undefined |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | X | X | 610 - 613 | Set mainframe channel speed: address = 100, data = 100 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | X | X | 650 - 653 | Undefined |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | X | X | 710 - 713 | Set mainframe channel speed: address = 100, data = 200 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | X | X | 750 - 753 | Set mainframe channel speed: address = 200, data = 200 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | X | X | 614 - 617 | Set SSD channel speed: address = 100, data = 100 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | X | X | 654 - 657 | Undefined |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | X | X | 714 - 717 | Set SSD channel speed: address = 100, data = 200 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | X | X | 754 - 757 | Set SSD channel speed: address = 200, data = 200 |

X = 0 or 1

## Channel Speed Selection Functions

A HISP channel instruction with modifier bits M2 and M3 both equal to 1 sets the operating speed of the mainframe or SSD channels. Bit $2^2$ of the *d* field or B register selects either the mainframe or SSD channels; both the input and the output channels are programmed at the same time.

Modifier bit M0 controls the speed of the address channels; modifier bit M1 controls the speed of the data channels. The address and data channels can be programmed independently to operate at either 100 Mbytes/s or 200 Mbytes/s. However, if the address channels are programmed to operate at 200 Mbytes/s, the data channels must also be programmed to operate at 200 Mbytes/s.

Select the speed of the HISP channels only during IOS initialization. Master clear commands and channel-clearing instructions do not affect the speed selection; therefore, it is not necessary to re-specify the channel speeds during normal IOS operation.

## Input Channel Functions

A HISP channel instruction with modifier bits M2 and M3 both equal to 0 and modifier bit M0 equal to 0 is a HISP input channel function. The following subsections describe each of the input functions in detail. Table 4-5 lists the functions.

Table 4-5. HISP Input Channel Functions

| Function | Description |
|---|---|
| HIA:0 | Clear channel |
| HIA:2 | Write upper starting address |
| HIA:3 | Write lower starting address |
| HIA:4 | Reserved for upper word count |
| HIA:5 | Write word count and start transfer |
| HIA:6 | Clear interrupt enable flag |
| HIA:7 | Set interrupt enable flag |
| HIA:12 | Read syndrome |
| HIA:13 | Read status |
| HIA:16 | Select buffer and pointer |
| HIA:17 | Set diagnostic mode |

### HIA:0 - Clear Channel

This function aborts any data transfer in progress, clears the channel busy and done flags, and clears the status register. To ensure that all activity has stopped on a channel, send an HIA:0 function to both subchannels. This function uses modifier bits M0 through M3.

### HIA:2 - Write Upper Starting Address

This function transfers the contents of the accumulator to the upper 16 bits of the starting address register. This function uses modifier bits M0 through M3.

HIA:3 - Write Lower Starting Address

This function transfers the contents of the accumulator to the lower 16 bits of the address register. This function uses modifier bits M0 through M3.

HIA:4 - Reserved for Upper Word Count

This function is reserved for expansion of the transfer word count register. It performs no operation. This function uses modifier bits M0 through M3.

HIA:5 - Write Word Count and Start Transfer

This function transfers the contents of the accumulator to the transfer word count register, sets the channel busy flag, and clears the channel done flag. If no transfer using the other subchannel is in progress, a data transfer starts immediately. If a transfer using the other subchannel is in progress, the data transfer begins when the other subchannel completes its transfer. This function uses modifier bits M0 through M3.

Do not use this function to start a transfer to an I/O buffer where a HISP channel transfer is in progress (except for a transfer using the other subchannel); unpredictable results will occur.

**Note:** The maximum word count is 65,535. A word count of 0 produces unpredictable results.

**Note:** Because of restrictions in the SSD, all data transfers to the SSD are limited to multiples of 64-word blocks.

HIA:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag. This function uses modifier bits M0 through M3.

HIA:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag. This function uses modifier bits M0 through M3.

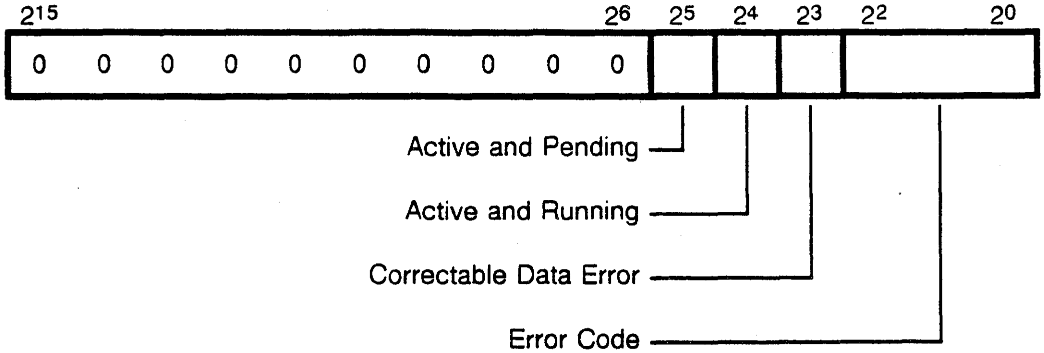
HIA:12 - Read Syndrome

This function transfers a SECDED syndrome to bits $2^0$ through $2^7$ of the accumulator. If no SECDED error has occurred since the last HIA:0 function, this function returns a syndrome of 0. If only uncorrectable errors have occurred, this function returns the syndrome of the first uncorrectable error. If correctable and uncorrectable errors have occurred, this function returns the syndrome of the first correctable error. This function uses modifier bits M0 through M3.

## HIA:13 - Read Status

This function transfers the contents of the channel status register to the accumulator.

Figure 4-11 shows the contents of the status register. Bits $2^0$ through $2^3$ contains error information. Bits $2^0$ through $2^2$ contain encoded error flags. (Refer to the "HISP Channel Errors" subsection in this section for more information on HISP errors.) Bit $2^3$ indicates that a correctable SECDED error occurred on the channel.



| $2^2$ | $2^1$ | $2^0$ | Error |
|---|---|---|---|
| 0 | 0 | 0 | No Error or Correctable Data Error |
| 0 | 0 | 1 | Word-count Error |
| 0 | 1 | 0 | Uncorrectable Data Error in the Mainframe or SSD |
| 0 | 1 | 1 | Address Error |
| 1 | 0 | 0 | Uncorrectable Data Error in IOS |
| 1 | 0 | 1 | Active Error |
| 1 | 1 | 0 | Reserved |
| 1 | 1 | 1 | Reserved |

Figure 4-11. HIA:13 Status Parcel

Status register bits $2^4$ and $2^5$ show the operating status of the subchannel. Bit $2^4$ (active and running) is set if a data transfer is in progress. Bit $2^5$ is set if a data transfer has been initiated (by an HIA:5 function), but the transfer is waiting for the other subchannel to complete a transfer. This function uses modifier bits M0 through M3.

## HIA:16 - Select Buffer and Pointer

This function uses the accumulator to select a buffer and pointer for a data transfer. Figure 4-12 shows the function of each accumulator bit. This function uses modifier bits M0 through M3.

## HIA:17 - Set Diagnostic Mode

This function uses accumulator bits $2^0$ through $2^2$ to select an operating mode for diagnostic purposes. The selected mode remains in effect until it is changed by another HIA:17 function. Figure 4-13 shows the function of each accumulator bit. The following list describes each diagnostic mode. This function uses modifier bits M0 through M3.

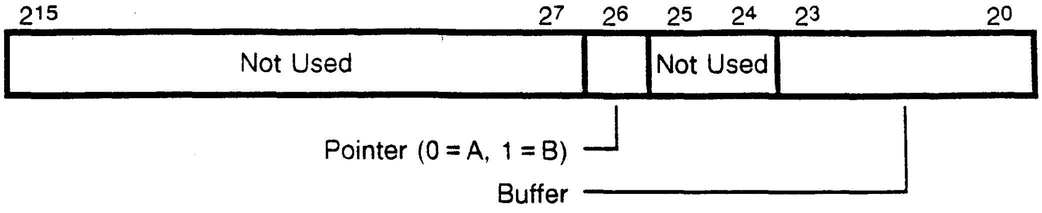
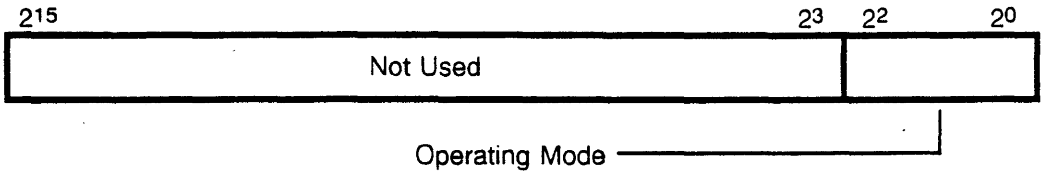


Figure 4-12. HIA:16 Control Parcel



| $2^2$ | $2^1$ | $2^0$ | Mode |
|---|---|---|---|
| 0 | 0 | 0 | Normal Operating Mode |
| 0 | 0 | 1 | Force Last Word Flag |
| 0 | 1 | 0 | Disable Last Word Flag |
| 0 | 1 | 1 | Force Address Parity Error |
| 1 | 0 | 0 | Disable One Address Ready Signal |
| 1 | 0 | 1 | Add Extra Address Ready Signal |
| 1 | 1 | 0 | Force Clear Channel |
| 1 | 1 | 1 | Disable Error Correction |

Figure 4-13. HIA:17 Control Parcel

Mode — Description

Force last word flag
Sets the last word flag in the channel control logic. If the transfer word count register is not 0, this mode creates a word count error.

Disable last word flag
Prevents the last word flag from setting. This mode creates a word count error when the word count register decrements to 0.

Force address parity error mode
Clears the parity bits of the first address parcel and sets the parity bits of the third address parcel.

Disable one Address Ready signal
Prevents one Address Ready signal from being sent when a data transfer is being initialized. This mode causes an Unexpected Data Ready signal received error when the channel begins receiving data.

Add extra Address Ready signal
Sends an extra Address Ready signal when a data transfer is being initialized. This mode causes an unexpected address ready received error.

| Mode | Description |
|---|---|
| Force clear channel | Sets the Clear Channel signal. This function aborts a transfer in progress and may cause an error, depending on where the Clear Channel signal sets in the transfer sequence. |
| Disable error correction | Disables the SECDED logic between the input channel and the I/O buffer. This function prevents SECDED errors that occur on the channel from being detected or corrected; data and check bits from the channel are written directly to the I/O buffer. |

## Output Channel Functions

A HISP channel instruction with modifier bits M2 and M3 both equal to 0 and modifier bit M0 equal to 1 is a HISP output channel function. The following subsections describe each of the output functions in detail. Table 4-6 lists the functions.

Table 4-6. HISP Output Channel Functions

| Function | Description |
|---|---|
| HOA:0 | Clear channel |
| HOA:2 | Write upper starting address |
| HOA:3 | Write lower starting address |
| HOA:4 | Reserved for upper word count |
| HOA:5 | Write word count and start transfer |
| HOA:6 | Clear interrupt enable flag |
| HOA:7 | Set interrupt enable flag |
| HOA:12 | Read syndrome |
| HOA:13 | Read status |
| HOA:16 | Select buffer and pointer |
| HOA:17 | Set diagnostic mode |

### HOA:0 - Clear Channel

This function aborts any data transfer in progress, clears the channel busy and done flags, and clears the status register. To ensure that all activity has stopped on a channel, send an HOA:0 function to both subchannels. This function uses modifier bits M0 through M3.

### HOA:2 - Write Upper Starting Address

This function transfers the contents of the accumulator to the upper 16 bits of the Address register. This function uses modifier bits M0 through M3.

### HOA:3 - Write Lower Starting Address

This function transfers the contents of the accumulator to the lower 16 bits of the address register. This function uses modifier bits M0 through M3.

### HOA:4 - Reserved for Upper Word Count

This function is reserved for expansion of the transfer word count register. It performs no operation. This function uses modifier bits M0 through M3.

### HOA:5 - Write Word Count and Start Transfer

This function transfers the contents of the accumulator to the word count register, sets the channel busy flag, and clears the channel done flag. If no transfer using the other subchannel is in progress, a data transfer starts immediately. If a transfer using the other subchannel is in progress, the data transfer begins when the other subchannel completes its transfer. This function uses modifier bits M0 through M3.

**Note:** Do not use this function to start a transfer from an I/O buffer where a HISP channel transfer is in progress (except for a transfer using the other subchannel); unpredictable results will occur.

**Note:** Because of restrictions in the SSD, all data transfers to the SSD are limited to multiples of 64-word blocks.

### HOA:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag. This function uses modifier bits M0 through M3.

### HOA:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag. This function uses modifier bits M0 through M3.

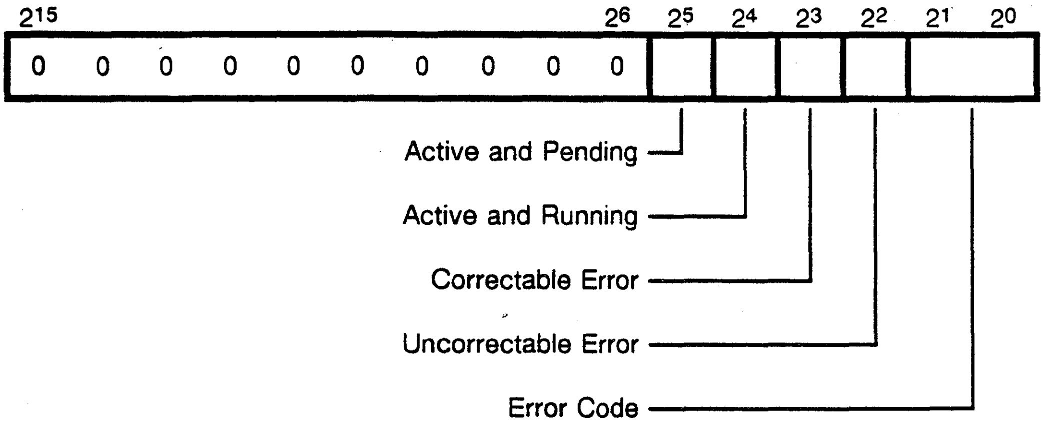
### HOA:12 - Read Syndrome

This function transfers a SECDED syndrome to bits $2^0$ through $2^7$ of the accumulator. If no SECDED error has occurred since the last HOA:0 function, this function returns a syndrome of 0. If only uncorrectable errors have occurred, this function returns the syndrome of the first uncorrectable error. If correctable and uncorrectable errors have occurred, this function returns the syndrome of the first correctable error. This function uses modifier bits M0 through M3.

HOA:13 - Read Status

This function transfers the contents of the channel status register to the accumulator.

Figure 4-14 shows the contents of the status register. Bits $2^0$ through $2^3$ contain error information. Bits $2^0$ and $2^1$ contain encoded error flags. (Refer to the "HISP Channel Errors" subsection in this section for more information on HISP channel errors.) Bits $2^2$ and $2^3$ indicate that an uncorrectable or correctable SECDED error occurred in the I/O buffer. This function uses modifier bits M0 through M3.



| $2^1$ | $2^0$ | Error |
|---|---|---|
| 0 | 0 | No Error or Correctable Data Error |
| 0 | 1 | Active Error |
| 1 | 0 | Uncorrectable Error in the Mainframe or SSD |
| 1 | 1 | Address Error |

Figure 4-14. HOA:13 Status Parcel

Status register bits $2^4$ and $2^5$ show the operating status of the subchannel. Bit $2^4$ (active and running) is set if a data transfer is in progress. Bit $2^5$ is set if a data transfer has been initiated by an HIA:5 function, but the transfer waits for the other subchannel to complete a transfer.

HOA:16 - Select Buffer and Pointer

This function uses the accumulator to select a buffer and a pointer for a data transfer. Figure 4-15 shows the function of each accumulator bit. This function uses modifier bits M0 through M3.

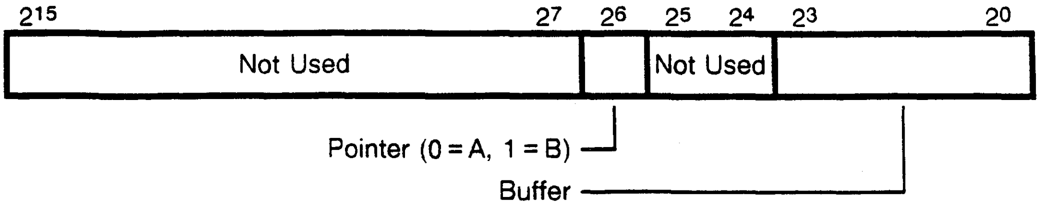
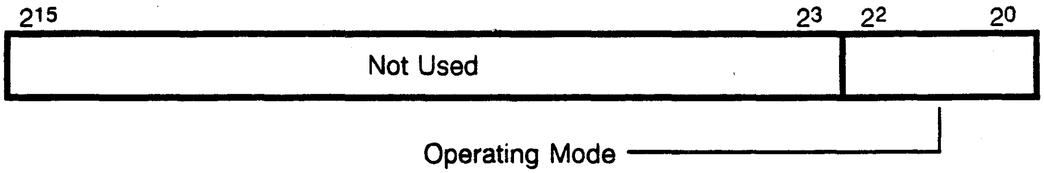


Figure 4-15. HOA:16 Control Parcel

HOA:17 - Set Diagnostic Mode

This function uses accumulator bits $2^0$ through $2^2$ to select an operating mode for diagnostic purposes. The selected mode remains in effect until changed by another HOA:17 function. Figure 4-16 shows the function of each accumulator bit. The following paragraphs describe each diagnostic mode. This function uses modifier bits M0 through M3.



| $2^2$ | $2^1$ | $2^0$ | Mode |
|---|---|---|---|
| 0 | 0 | 0 | Normal Operating Mode |
| 0 | 0 | 1 | Force Last Word Flag |
| 0 | 1 | 0 | Disable Last Word Flag |
| 0 | 1 | 1 | Force Address Parity Error |
| 1 | 0 | 0 | Disable One Address Ready Signal |
| 1 | 0 | 1 | Add Extra Address Ready Signal |
| 1 | 1 | 0 | Force Clear Channel |
| 1 | 1 | 1 | Disable Error Correction |

Figure 4-16. HOA:17 Control Parcel

| Mode | Description |
|---|---|
| Force last word flag | Sets the last word flag in the channel control logic. If the transfer word count register is not 0, this mode creates a word count error. |
| Disable last word mode flag | Prevents the last word flag from setting. The mode creates a word count error when the word count register decrements to 0. |
| Force address parity error mode | Clears the parity bits of the first address parcel and sets the parity bits of the third address parcel. |

| Mode | Description |
|---|---|
| Disable one Address Ready signal | Prevents one Address Ready signal from being sent when a data transfer is being initialized. This mode causes an unexpected data ready signal received error when the channel begins receiving data. |
| Add extra Address Ready | Sends an extra Address Ready signal when a data transfer signal is initiated. This mode causes an unexpected address ready received error. |
| Force clear channel | Sets the Clear Channel signal. This function aborts a transfer in progress and may cause an error, depending on where the Clear Channel signal sets in the transfer sequence. |
| Disable error-correction | Disables the SECDED logic in the I/O buffer and the SECDED logic in the mainframe or SSD. This function prevents SECDED errors that occur in the IOS or on the channel from being detected or corrected; data and check bits from the I/O buffer are sent directly through the channel and are written directly to mainframe or SSD memory. |

# CONTENTS

## FIGURES

**FIGURES (continued)**

**TABLES**

# 5. CHANNEL ADAPTERS

Channel adapters enable the I/O subsystem (IOS) to communicate with external devices such as disk drives, tape drives, and front-end interfaces (FEIs). There are several different types of channel adapters; refer to Table 5-1 for a list of the channel adapters and the external devices they service. The following subsections contain information common to all channel adapters; information specific to each channel adapter is provided later in this section.

Each cluster contains 16 channel adapters, one for each I/O buffer. Each of the four auxiliary I/O processors (EIOPs) in the cluster controls four channel adapters.

Channel adapters receive input data from and send output data to external devices. A channel adapter receives input data in the format used by the external device. The channel adapter assembles the input data into a 64-bit word plus 8 bits of single-error correction/double-error detection (SECDED) logic. The channel adapter then sends the input data to the I/O buffer.

The channel adapter receives output data from the I/O buffer as a 64-bit word plus 8 SECDED bits. The channel adapter disassembles the output data into the format used by the external device; it then sends the data to the external device.

Some channel adapters (DCA-2 and TCA-1) support data transfers to and from local memory; this feature is used to transfer microcode to a disk controller.

Table 5-1. Channel Adapters

| Channel Adapter | External Device |
|---|---|
| CCA-1 | LOSP channel |
| DCA-1 | DD-39, DD-49, DD-40, and DD-41 disk drives |
| DCA-2 | DD-60 and DD-61 disk drives |
| HCA-3 | HIPPI input channel |
| HCA-4 | HIPPI output channel |
| TCA-1 | IBM block multiplexer channel |

# COMMON CHANNEL ADAPTER OPERATIONS

Two operations are common to all channel adapters: channel adapter programming and channel adapter data transfers. Refer to the following subsection for information on these two operations.

## Channel Adapter Programming

Channel adapter instructions consist of the channel function, function modifier bits, accumulator contents and channel numbers. The channel function, function modifier bits, and accumulator contents are sent to the channel adapter. The channel number is not sent to the channel adapter. Refer to Figure 5-1 for an illustration of the channel adapter instruction format.

Bits $2^0$ through $2^3$ of the *f* field of the instruction contain the channel function. The channel function specifies to the channel adapter which I/O function is being requested. Bits $2^4$ through $2^6$ are the instruction decode. Clearing bit $2^4$ selects the instruction *d* field; setting bit $2^4$ selects the B register. Bits $2^5$ and $2^6$ are always set for channel functions (instructions 140-177).

Bits $2^0$ through $2^4$ of the *d* field (instructions 140 through 157) or bits $2^0$ through $2^4$ of the B register (instructions 160 through 177) represent the channel number. Each EIOP uses the following channels to control the channel adapters:

- Channels 30 and 31 control channel adapter 0.
- Channels 32 and 33 control channel adapter 1.
- Channels 34 and 35 control channel adapter 2.
- Channels 36 and 37 control channel adapter 3.

Except for channels that access disk drives, tape drives, and HIPPI channels, even-numbered channels are input channels; odd-numbered channels are output channels.

The EIOP uses only the channel number to select one of four channel adapters; the EIOP does not send the channel number to the channel adapters. Therefore, the input or output data transfer direction must be communicated to the channel adapter logic in the M0 bit.

Bits $2^5$ through $2^8$ of the *d* field or of the B register contain 4 function modifier bits designated M0 through M3. M0, the least-significant bit, is set to indicate an output operation (odd channel number) or is cleared to indicate an input operation (even channel number). The purpose of bits M1 through M3 is dependent on the channel adapter type and channel function used.

## Channel Adapter Data Transfers

Each I/O buffer has two pointers (a and b) for channel adapter data transfers. The even channel uses the a pointer; the odd channel uses the b pointer. If data is to be read into the I/O buffer, the a pointer points to a designated address location, and data is read into that location. The a pointer then increments to the next designated address location. If data is to be read from the I/O buffer, the b pointer points to a designated address location, and data is read from that location. The b pointer then increments to the next

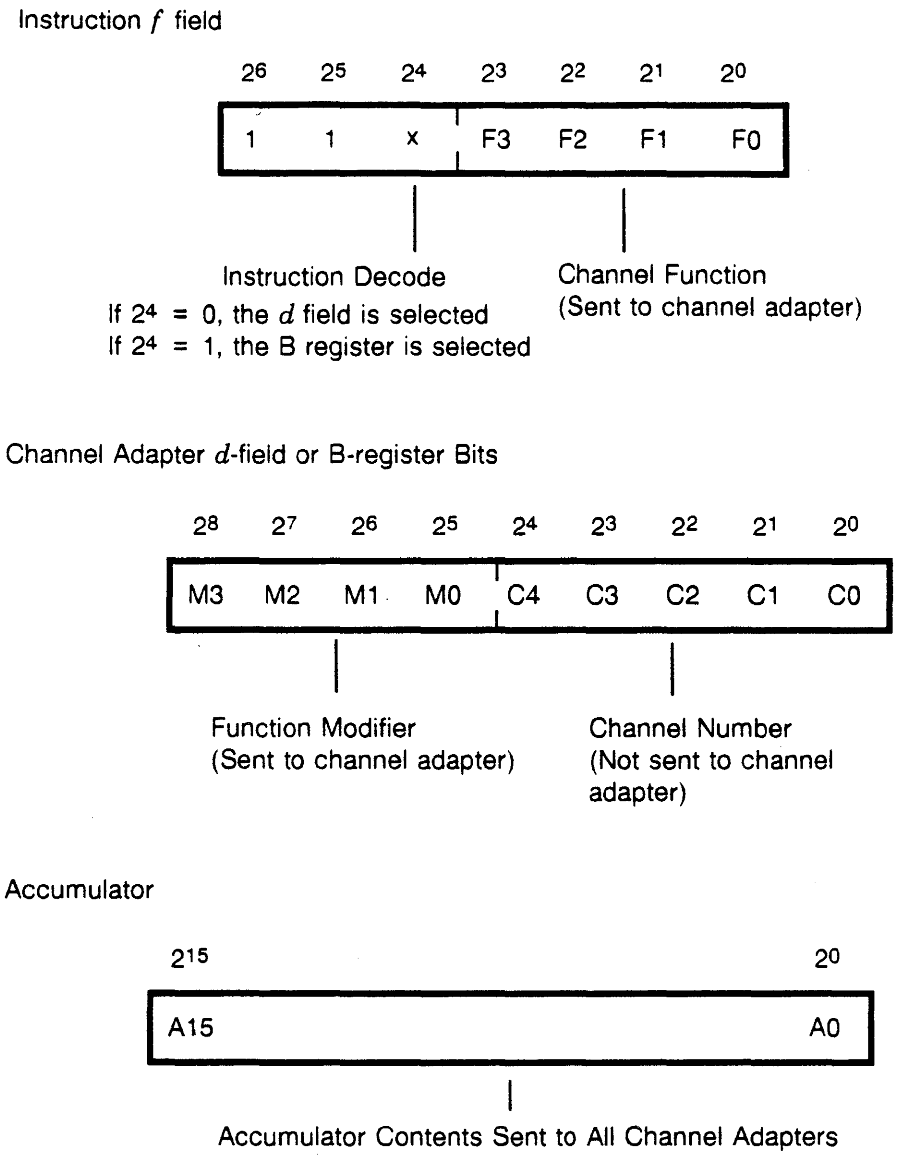

Instruction *f* field
$2^6$ $2^5$ $2^4$ $2^3$ $2^2$ $2^1$ $2^0$
1 1 x F3 F2 F1 F0
Instruction Decode
If $2^4$ = 0, the *d* field is selected
If $2^4$ = 1, the B register is selected
Channel Function
(Sent to channel adapter)
Channel Adapter *d*-field or B-register Bits
$2^8$ $2^7$ $2^6$ $2^5$ $2^4$ $2^3$ $2^2$ $2^1$ $2^0$
M3 M2 M1 M0 C4 C3 C2 C1 C0
Function Modifier
(Sent to channel adapter)
Channel Number
(Not sent to channel adapter)
Accumulator
$2^{15}$ $2^0$
A15 A0
Accumulator Contents Sent to All Channel Adapters

Figure 5-1. Channel Adapter Instruction Formats

designated address location. Channel adapter pointers a and b can also be decremented for support of tape read backwards commands. Refer to Section 3 of this manual for additional information about I/O buffer programming.

The channel adapter channels have busy, done, and interrupt enable flags that are read by a program. The busy flag indicates that a data transfer is in progress and the done flag indicates that the transfer is complete. Channel adapter channels generate interrupt requests when the done flag sets at the completion of a data transfer. The interrupt enable flag controls the ability of the channel to receive interrupts. If the flag is set, the channel will receive interrupts; if the flag is not set, the channel will not receive interrupts. Refer to Section 2 of this manual for more information about channel interrupts.

# CCA-1 CHANNEL ADAPTER

The CCA1 channel adapter enables the IOS-E to communicate with front-end interfaces (FEIs) through a low-speed (LOSP) channel pair. The LOSP channel pair consists of an input and an output channel. The even channel is the input channel; the odd channel is the output channel. The signal and control protocol is provided in the following subsections.

**Note:** The CCA-1 LOSP channels are not the same as the LOSP channels that are dedicated to the operator workstation (OWS) and maintenance workstation (MWS).

## Data Transfers

The CCA-1 transfers data to or from an external device in either a 6-Mbyte/s mode, a 12-Mbyte/s mode, or a 12-Mbyte/s device mode. These modes configure the hardware for the desired speed, or test the various modes in a loopback test for diagnostic purposes. The three modes of operation are explained in the following subsections. Refer also to the functions CIA:17 and COA:17 for more information on these modes.

The CCA-1 operates in a full-duplex mode, that is, data transfers can take place in both directions simultaneously. Each data transfer is 1 to 65,536 words long; each word is 64 bits plus 8 check bits. Data from the mainframe to the CCA-1 is SECDED protected; data from the CCA-1 to the external device is parity protected.

### 6-Mbyte/s Mode

Three signals control a 6-Mbyte/s data transfer: Ready, Resume, and Disconnect. Refer to Figure 5-2 for an illustration of the 6-Mbyte/s input channel signals.

**Note:** The CCA-1 input channel, when operating in the 6-Mbyte/s mode, can either hang or receive bad data if the channel is activated at the same time a Ready signal is received. The following sequence of instructions replaces the input channel activation command and prevents these errors.

1. Disable system interrupts (if enabled)
2. Store accumulator contents to EIOP local memory
3. Load accumulator contents from EIOP local memory
4. Perform a CIA:5 function (write parcel count and start transfer)
5. Enable system interrupts

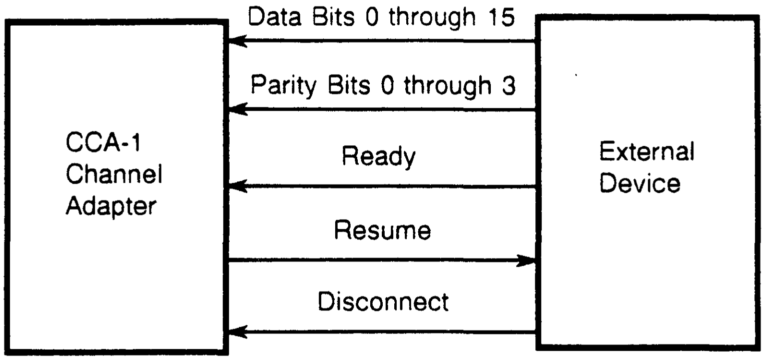


Figure 5-2. CCA-1 Input Channel Signals

For external device-to-CCA-1 data transfers, the control sequence is as follows:

1. The external device sends the Ready and Disconnect signals to the CCA-1.
2. The CCA-1 sends the Resume signal to the external device.
3. The external device places one parcel of data and four parity bits on the channel. The external device then activates the Ready signal which informs the CCA-1 that data is waiting on the channel.
4. The CCA-1 reads the data and parity bits from the channel and checks for parity errors. The CCA-1 then activates the Resume signal to indicate it has received the data.
5. Steps 3 and 4 are repeated until all data is transferred across the channel.
6. The external device activates the Disconnect signal to indicate the transfer is complete.

For CCA-1-to-external device transfers, the same control signals and control sequence apply, but the directions of the control signals are reversed. The CCA-1 sends the Ready and Disconnect signals to the external device; the external device sends the Resume signal to the CCA-1. Refer to Figure 5-3 for an illustration of the 6-Mbyte/s output channel signals.

### 12-Mbyte/s Mode

Three signals control a 12-Mbyte/s data transfer: Ready, Resume, and Disconnect. For external device-to-CCA-1 data transfers, the control sequence is as follows:

1. The external device sends the Ready and Disconnect signals to the CCA-1.
2. The CCA-1 sends four Resume signals to the external device.
3. The external device responds with four parcels of data and four Ready signals (one Ready signal for each parcel of data).

4. Steps 2 and 3 are repeated until all data is transferred across the channel.
5. The external device activates the Disconnect signal to indicate the transfer is complete.

For CCA-1-to-external device transfers, the same control signals apply, but the control sequence and the directions of the control signals are different. The control sequence is as follows:

1. The CCA-1 sends four parcels of data and four Ready signals to the external device (one Ready signal for each parcel of data).
2. The external device responds with one Resume signal.
3. Steps 1 and 2 are repeated until all data is transferred across the channel.
4. The CCA-1 activates the Disconnect signal to indicate the transfer is complete.

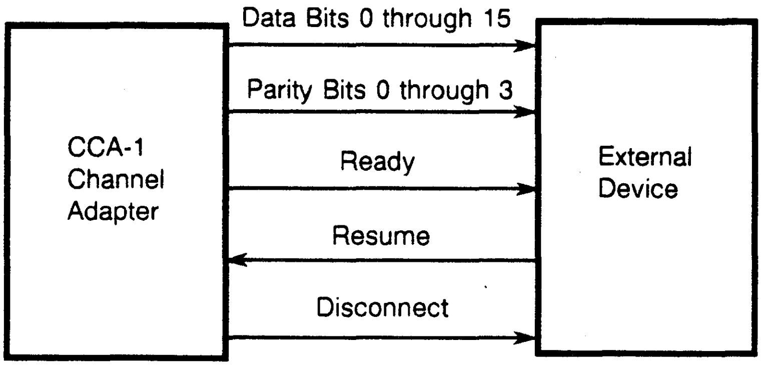


Figure 5-3. CCA-1 Output Channel Signals

### 12-Mbyte/s Device Mode

The 12-Mbyte/s device mode operates in the same way as the 12-Mbyte/s mode. This mode allows two channel adapters to be configured together for maintenance purposes. Refer to the functions CIA:17 and COA:17 for more information on this mode.

## Data Parity Errors

Parity error flags indicate that a parity error has occurred. When a parity error is detected, the parity error flag sets, the transfer is completed, and an interrupt request is issued.

Data parity errors that occur during the data portion of a transfer are detected by the EIOP input channel. The status response to the EIOP provides the parity error information. (Refer to input channel function CIA:13 later in this section for information on the status register.)

## More-data-on-channel and Ready-waiting Flags

The more-data-on-channel flag sets when the parcel count becomes 0 and the channel adapter receives another Ready signal.

The ready-waiting flag sets when the channel adapter receives a Ready signal from an external device while the channel is inactive.

## Input Channel Functions

The CCA-1 input channel functions control data flow from an external device, to a channel adapter, and then to an I/O buffer. Refer to Table 5-2 for a list of the CCA-1 input functions.

Table 5-2. CCA-1 Input Channel Functions

| Function | Description |
|---|---|
| CIA:0 | Clear channel |
| CIA:3 | Clear error flags |
| CIA:4 | Clear channel flags and input buffer pointer |
| CIA:5 | Write parcel count and start transfer |
| CIA:6 | Clear interrupt enable flag |
| CIA:7 | Set interrupt enable flag |
| CIA:12 | Read channel parcel count |
| CIA:13 | Read status |
| CIA:17 | Write modes |

### CIA:0 - Clear Channel

This function aborts any transfer in progress and clears the channel busy and done flags.

### CIA:3 - Clear Error Flags

This function clears the input channel parity error flags in the status register.

### CIA:4 - Clear Channel Flags and Input Buffer Pointer

This function clears the channel flags and input buffer pointer.

### CIA:5 - Write Parcel Count and Start Transfer

This function transfers the contents of the accumulator into the channel adapter parcel count register and starts the transfer of data to the I/O buffer. The parcel count is limited to four parcel boundaries; therefore, the two least significant bits of the accumulator content must be 0's.

The parcel counter decrements as each parcel is received. When a Disconnect signal is received by the channel adapter, the channel busy flag clears and the channel done flag sets. If the parcel count is 0, and another Ready signal is received by the channel adapter, the channel busy flag clears and the more-data-on-channel flag sets.

### CIA:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag.

### CIA:7 - Set Interrupt Enable Flag

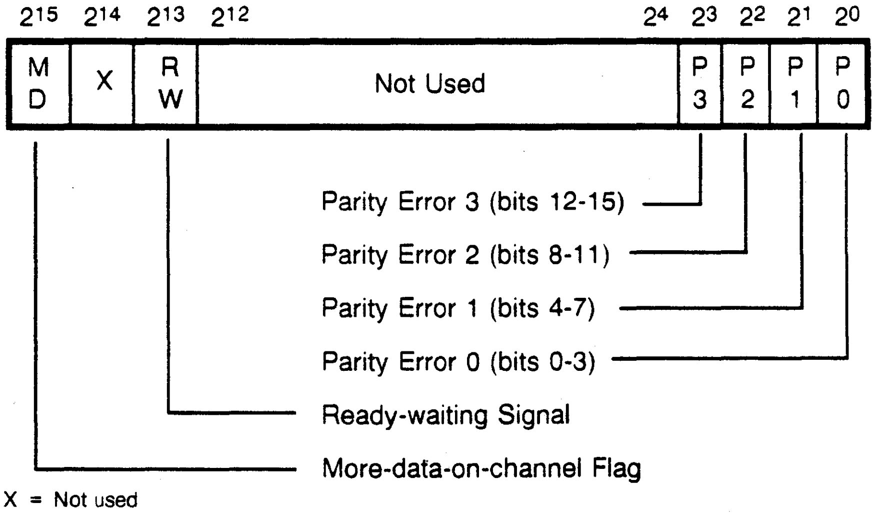
This function sets the interrupt enable flag.

### CIA:12 - Read Channel Parcel Count

This function transfers the channel parcel count to the accumulator. If a transfer is in progress when the parcel count is read, the channel parcel count may not be accurate.

### CIA:13 - Read Status

This function reads the error status register and sends the error status register contents to the accumulator. Refer to Figure 5-4 for an illustration of the LOSP channel CIA:13 status parcel. Bits $2^0$ through $2^3$, when set, indicate parity errors for bits 0 through 15.



2^15 2^14 2^13 2^12
2^4 2^3 2^2 2^1 2^0
M D
X
R W
Not Used
P 3
P 2
P 1
P 0
Parity Error 3 (bits 12-15)
Parity Error 2 (bits 8-11)
Parity Error 1 (bits 4-7)
Parity Error 0 (bits 0-3)
Ready-waiting Signal
More-data-on-channel Flag
X = Not used

Figure 5-4. Channel CIA:13 Status Parcel

Bits $2^{13}$ and $2^{15}$ set when a Ready-waiting signal is received or the more-data-on-channel flag sets. Bits $2^4$ through $2^{12}$ and bit $2^{14}$ are not used.

### CIA:17 - Write Modes

This function configures the channel hardware to the desired operating speed. This function can also be used to test the various mode select bits in a loopback test.

Refer to Figure 5-5 for an illustration of the LOSP channel CIA:17 mode select bits. The 6-Mbyte/s mode is selected by setting bit $2^0$; the 12-Mbyte/s mode is selected by setting bit $2^1$; the 12-Mbyte/s device modes is selected by setting bit $2^2$. Bits $2^3$ through $2^{15}$ are not used. Only one mode can be selected at a time. If more than one mode is simultaneously selected, unpredictable results will occur.

The 6-Mbyte/s and 12-Mbyte/s modes are described in the "Data Transfers" subsection provided earlier in this section. The 12-Mbyte/s device mode allows two channel adapters to be configured together for maintenance purposes.

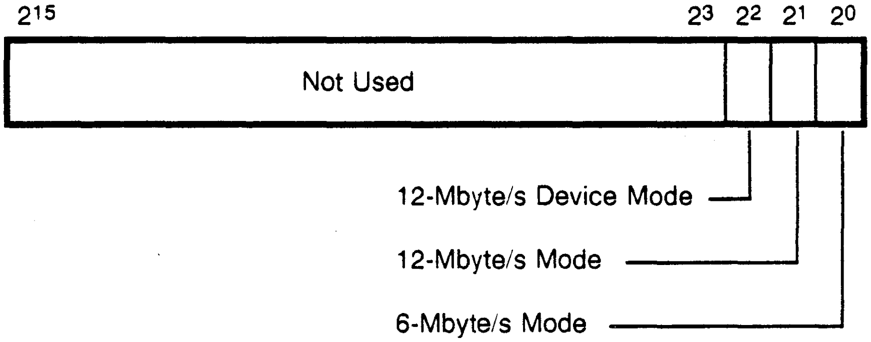


Figure 5-5. Channel CIA:17 Mode Select Bits

## Output Channel Functions

The CCA-1 output channel functions control data flow from an I/O buffer, to a channel adapter, and then to an external device. Refer to Table 5-3 for a list of the CCA-1 output functions.

Table 5-3. CCA-1 Output Channel Functions

| Function | Description |
|---|---|
| COA:0 | Clear channel |
| COA:3 | Clear error flags |
| COA:4 | Set/clear external signals |
| COA:5 | Write parcel count and start transfer |
| COA:6 | Clear interrupt enable flag |
| COA:7 | Set interrupt enable flag |
| COA:12 | Read channel parcel count |
| COA:13 | Read status |
| COA:17 | Write modes |

### COA:0 - Clear Channel

This function aborts any transfer in progress and clears the channel busy and channel done flags. This function uses modifier bit M0.

### COA:3 - Clear Error Flags

This function clears the output data SECDED error flags in the status register. This function uses modifier bit M0.

### COA:4 - Set/Clear External Signals

This function uses accumulator bits $2^8$, $2^9$, and $2^{14}$ to select various external signal options (refer to Figure 5-6 for an illustration of the LOSP output channel COA:4 bits). Bits $2^0$ through $2^7$, $2^{10}$ through $2^{13}$, and bit $2^{15}$ are not used. This function uses modifier bit M0.

Setting bit $2^8$ sends a Disconnect signal to an external device when necessary for program control. The Disconnect signal is sent regardless of the state of bit $2^9$. The Disconnect signal is cleared by using function COA:4 with bit $2^8$ cleared.

A 6-Mbyte/s LOSP mode transfer can be terminated by sending a Disconnect signal after the last block of data has been sent (using function COA:4 with bit $2^8$ set). The transfer

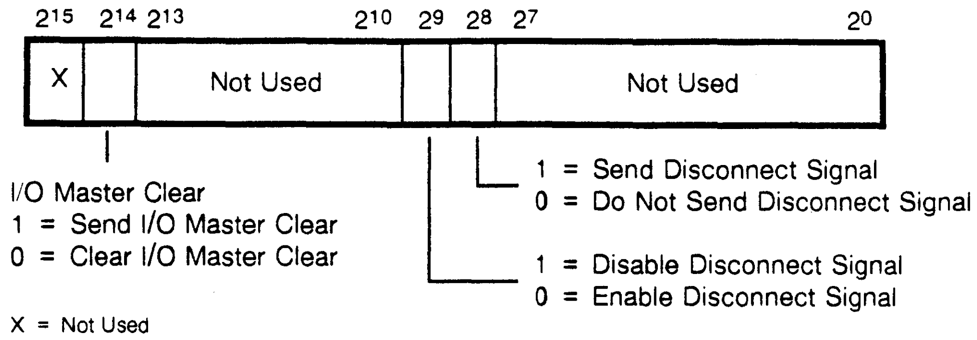

$2^{15}$
$2^{14}$
$2^{13}$
$2^{10}$
$2^{9}$
$2^{8}$
$2^{7}$
$2^{0}$
X
Not Used
Not Used
I/O Master Clear
1 = Send I/O Master Clear
0 = Clear I/O Master Clear
X = Not Used
1 = Send Disconnect Signal
0 = Do Not Send Disconnect Signal
1 = Disable Disconnect Signal
0 = Enable Disconnect Signal

Figure 5-6. COA:4 Control Bits

can also be terminated by enabling disconnects before sending the last block of data (using function COA:4 with bit $2^9$ cleared).

**Note:** Do not terminate a 12-Mbyte/s HISP mode transfer by sending a Disconnect signal after the last block of data has been sent. The 12-Mbyte/s protocol calls for sending a Disconnect signal along with the final word.

Setting bit $2^9$ disables disconnects and allows large blocks of data to be sent as a series of smaller blocks without intermediate Disconnect signals. Clearing bit $2^9$ re-enables the Disconnect signal.

Bit $2^{14}$ controls the I/O Master Clear signal on the input and output channels. Setting bit $2^{14}$ sends the Master Clear signal; this signal continues until bit $2^{14}$ is cleared.

### COA:5 - Write Parcel Count and Start Transfer

This function transfers the contents of the accumulator into the parcel count register and starts the transfer. This function uses modifier bit M0.

The output channel in 6-Mbyte/s LOSP mode sends an odd number of parcels and does not have to observe 4-parcel Cray word boundaries. However, if a large transfer is sent as a series of smaller data blocks, all but the last block must observe Cray word boundaries.

When using the 12-Mbyte/s HISP and 12-Mbyte/s device modes, the parcel count must end the transfer on a 4-parcel word boundary.

### COA:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag. This function uses modifier bit M0.

### COA:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag. This function uses modifier bit M0.

### COA:12 - Read Channel Parcel Count

This function reads the channel parcel count and transfers the parcel count to the accumulator. If a transfer is in progress when the parcel count is read, the channel parcel count may not be accurate. This function uses modifier bit M0.

### COA:13 - Read Status

This function reads the error status register and sends the error status register contents to the accumulator. Refer to Figure 5-7 for an illustration of the LOSP output channel COA:13 status bits. This function uses modifier bit M0.

| $2^{15}$ $2^{12}$ | $2^{11}$ $2^{8}$ | $2^{7}$ $2^{0}$ |
|---|---|---|
| Double-bit Error Count | Single-bit Error Count | Syndrome |

Figure 5-7. COA:13 Status Bits

Bits $2^0$ through $2^{15}$ contain the LOSP output channel status bits. Bits $2^0$ through $2^7$ hold the syndrome. The first single-bit error to occur after clearing the status-bit register is entered into the syndrome-bit register. When a single-bit error syndrome is entered into the register, the single-bit error syndrome cannot be overridden by a double-bit error syndrome, or by another single-bit error syndrome. If there are no single-bit errors, the most recent double-bit error is registered.

If a double-bit error occurs before a single-bit error, the double-bit error syndrome is entered into the syndrome-bit register. The double-bit error remains in the syndrome-bit register until a single-bit error occurs.

Bits $2^8$ through $2^{11}$ hold the single-bit error count. The single-bit error count is the number of single-bit errors that have occurred since the status register was last cleared (a COA:3 function clears the status register).

Bits $2^{12}$ through $2^{15}$ hold the double-bit error count. The double-bit error count is the number of double-bit errors that have occurred since the status register was last cleared.

### COA:17 - Write Modes

This function configures the channel hardware to the desired operating speed and provides two options for the handling of double-bit errors. This function uses modifier bit M0.

Refer to Figure 5-8 for an illustration of the output channel COA:17 mode bits. Bits $2^0$ through $2^2$ select the 6-Mbyte/s, 12-Mbyte/s, or 12-Mbyte/s device mode. Bits $2^4$ through $2^{15}$ are not used. Only one mode can be selected at a time. If more than one mode is simultaneously selected, unpredictable programming results will occur.

Bit $2^3$ detects double-bit errors and, when set, stops a transfer if a double-bit error occurs. Bit $2^3$, when cleared, allows a transfer to continue if a double-bit error occurs.

The 6-Mbyte/s and 12-Mbyte/s modes are described in the "Data Transfers" subsection provided earlier in this section. The 12-Mbyte/s device mode allows two channel adapters to be configured together for maintenance purposes.

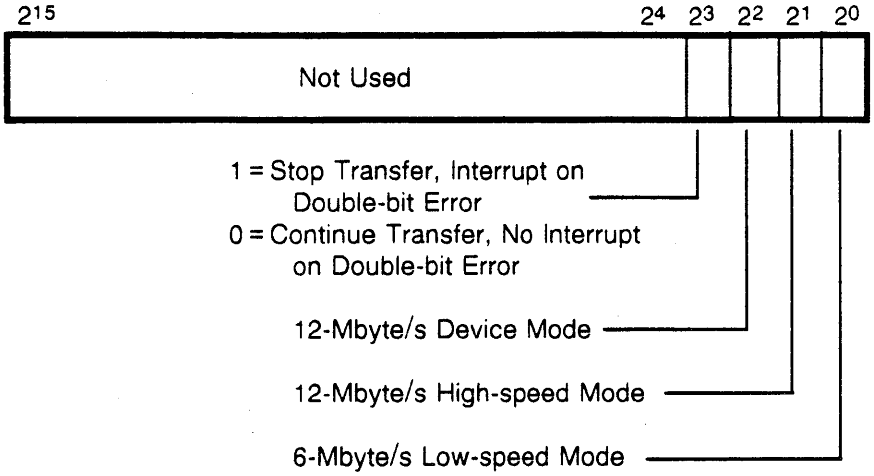

$2^{15}$
$2^{4}$
$2^{3}$
$2^{2}$
$2^{1}$
$2^{0}$
Not Used
1 = Stop Transfer, Interrupt on Double-bit Error
0 = Continue Transfer, No Interrupt on Double-bit Error
12-Mbyte/s Device Mode
12-Mbyte/s High-speed Mode
6-Mbyte/s Low-speed Mode

Figure 5-8. Output Channel COA:17 Mode Bits

# DCA-1 CHANNEL ADAPTER

The DCA-1 disk channel adapter enables the IOS-E to communicate with disk storage units. Specifically, the DCA-1 controls transfers between an EIOP and DD-39, DD-49, DD-40, and DD-41 disk drives. Each DCA-1 can control one DD-39 or one DD-49 and can control up to two DD-40 or two DD-41s.

## Data Transfers

The DCA-1 transfers data to or from an external device at a rate of 12 Mbytes/s. The DCA-1 operates in a half-duplex mode; that is, data transfers can take place on either channel in only one direction at a time. Each sector transfer is 1 to 512 words long. Data between the mainframe and the DCA-1 is SECDED protected; data from the DCA-1 to the external device is parity protected.

## Channel Signals

Refer to Figure 5-9 for an illustration of the DCA-1 channel signals. The DCA-1 channel signals, and channel signal definitions are provided in the following list.

| Signal | Definition |
|---|---|
| Write Clock | The Write Clock signal is generated by the channel adapter for the synchronization of commands and data to the external device. |
| Function/Data Ready | The Function/Data Ready signal is active during bus-out cycles that carry a valid function on the function code lines. |
| Function Code 0 through 3 | The Function Code signal carries the function to be performed by the drive. |
| Function Parity Code | The Function Parity Code signal carries odd parity of function code bits $2^0$ through $2^3$. Parity is checked only on bus out cycles when the Function/Data Ready signal is active. |
| Bus Out 0 through 15 | The Bus Out 0 through 15 signals are a 16 bit data bus sent from the EIOP to the external device. Bus Out 0 is data bit $2^0$ and Bus out 15 is data bit $2^{15}$. |
| Bus Out Parity | The Bus Out Parity signal carries odd parity of Bus Out 0 through 15. Parity is only checked on Bus-out cycles when the Function/Data Ready signal is active. |
| Read Clock | The Read Clock signal is generated by the external device for the synchronization of commands and data to the channel adapter. |

Signal — Definition

Status/Data Ready — The Status/Data Ready signal is active during bus-in cycles that carry a valid function on the function code lines.

Error — The Error signal is used in conjunction with the Done signal to indicate that at least one error condition occurred during the current function.

Done — The Done signal indicates the completion of a command.

Ready — The Ready signal indicates the availability of the external device to accept interface commands.

Index/Sector Mark — The Index/Sector Mark signal carries an encoded index and sector mark from the external device to the channel adapter.

Parity/Error — The Parity/Error signal carries odd parity of status/data ready, error, done, and ready. Parity is valid on bus-in cycles when the Ready signal is active.

Bus In 0 through 15 — The Bus In 0 through 15 signals are a 16 bit data bus sent from the external device to the EIOP. Bus In 0 is data bit $2^0$ and Bus In 15 is data bit $2^{15}$.

Bus In Parity — The Bus In Parity signal carries odd parity of Bus In 0 through 15. Parity is only checked on Bus-in cycles when the Status/Data Ready signal is active.

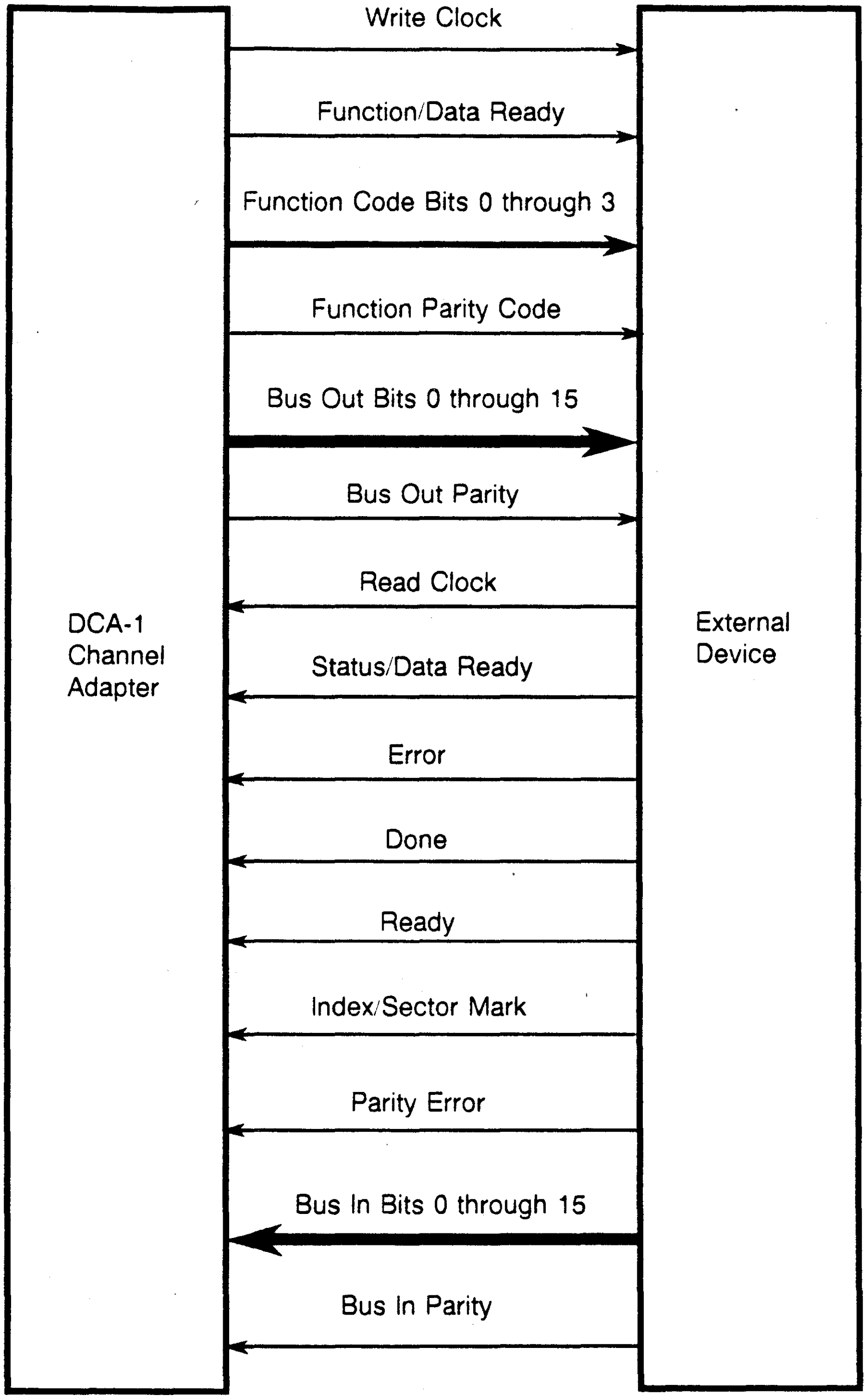


Figure 5-9. DCA-1 Channel Signals

## Channel Functions

Refer to Table 5-4 for a list of the DCA-1 disk channel functions.

Table 5-4. DCA-1 Disk Channel Functions

| Function | Description |
|---|---|
| DCA1:0 | Clear channel |
| DCA1:4 | Wait angular position interrupt |
| DCA1:5 | Select disk function and transfer length |
| DCA1:6 | Clear interrupt enable flag |
| DCA1:7 | Set interrupt enable flag |
| DCA1:12 | Read status 0 |
| DCA1:13 | Read status 1 |
| DCA1:16 | Write disk channel parameter |
| DCA1:17 | Write channel adapter modes |

### DCA1:0 - Clear Channel

This function clears the channel busy and channel done flags, clears the device lost ready status bit (refer to function DCA1:13), and clears the status register. This function uses modifier bit M0.

### DCA1:4 - Wait Angular Position Interrupt

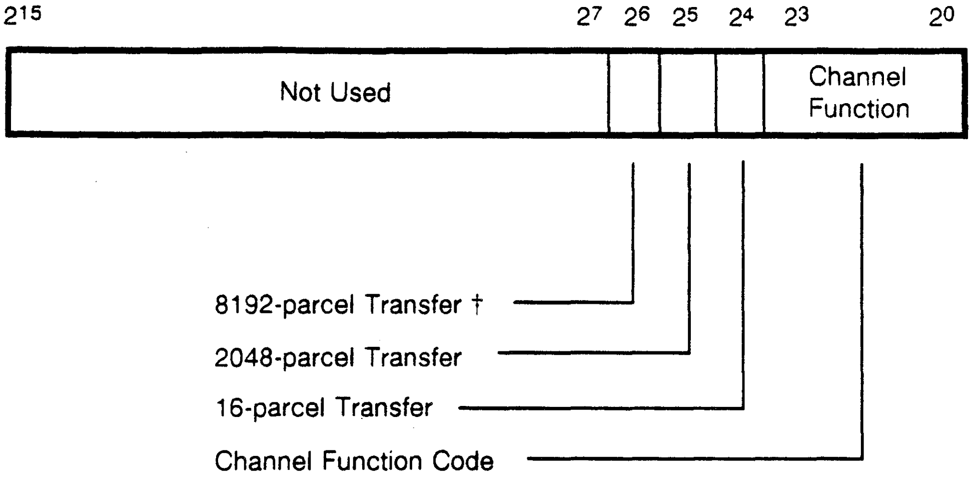
This function enables the next immediate sector mark or index mark of a disk. The angular position of the disk can then be read by using a DCA:13 Read Status 1 function on the odd-numbered channel. This function uses modifier bit M0.

### DCA1:5 - Select Disk Function and Transfer Length

The DCA:5 function transmits a device function, and clears the a or b status registers. Refer to Figure 5-10 for the accumulator bit definitions. The value in the accumulator specifies which type of function is being requested. The parameter previously sent by a DCA:16 function provides the bus-out data. This function uses modifier bit M0.

### DCA1:6 - Clear Interrupt Enable Flag

This function clears the channel interrupt enable flag. This function uses modifier bit M0.



† Reserved for future use.

Channel Function Definitions

| | |
|---|---|
| 0000 | Echo |
| 0001 | Select |
| 0010 | Read |
| 0011 | Write |
| 0100 | Head Select |
| 0101 | Cylinder Select |
| 0110 | Data Transfer |
| 0111 | Select Status |
| 1000 | General Status |
| 1001 | Diagnostic |
| 1010 | Not Used |
| 1011 | Reset |
| 1100 | Clear Faults |
| 1101 | Return to 0 |
| 1110 | Release Opposite Channel and Select |
| 1111 | Release |

Figure 5-10. DCA1:5 Function Accumulator Bits

### DCA1:7 - Set Interrupt Enable Flag

This function sets the channel interrupt enable flag. This function uses modifier bit M0.

### DCA1:12 - Read Status 0

This function returns two separate status parcels to the accumulator. This function uses modifier bit M0. Refer to Table 5-5 for descriptions of the DCA1:12 status parcels.

Issuing this function on an even-numbered channel returns the a status. Issuing it on an odd-numbered channel returns the b status. Each status register holds the two parcels of status bits with bit assignments as discussed in the following paragraph.

Table 5-5. DCA1:12 Status Parcels

| Status Bit | Description | |
|---|---|---|
| | Accumulator Bit = 0 | Accumulator Bit = 1 |
| $2^0$ | Drive ready † | First syndrome, bit $2^0$ |
| $2^1$ | Status/data ready † | First syndrome, bit $2^1$ |
| $2^2$ | Done † | First syndrome, bit $2^2$ |
| $2^3$ | Drive error † | First syndrome, bit $2^3$ |
| $2^4$ | Control parity error † | First syndrome, bit $2^4$ |
| $2^5$ | Status parity error † | First syndrome, bit $2^5$ |
| $2^6$ | Input data parity error † | First syndrome, bit $2^6$ |
| $2^7$ | Data transfer incomplete † | First syndrome, bit $2^7$ |
| $2^8$ | Single bit SECDED error † | Single-bit error count bit $2^0$ |
| $2^9$ | Double bit SECDED error † | Single-bit error count bit $2^1$ |
| $2^{10}$ | Not used | Single-bit error count bit $2^2$ |
| $2^{11}$ | Not used | Single-bit error count bit $2^3$ |
| $2^{12}$ | Not used | Double-bit error count bit $2^0$ |
| $2^{13}$ | Not used | Double-bit error count bit $2^1$ |
| $2^{14}$ | Device lost ready | Double-bit error count bit $2^2$ |
| $2^{15}$ | Device ready for functions | Double-bit error count bit $2^3$ ‡ |

† Status bits $2^0$ through $2^9$ are ending status conditions when the done flag is received from the device.

‡ The double-bit error count stops at $17_8$.

The first syndrome byte is loaded when an error occurs, and is not changed until the register is cleared. If the first error is a double-bit error, that syndrome is stored in the status register. Then if a single-bit error occurs, the single-bit error syndrome replaces the double-bit error syndrome in the status. As more single-bit errors occur, the first single-bit error syndrome that is stored is not replaced. Instead, the single bit error count is incremented to show the total number of single-bit errors that have occurred since the status register was last cleared. Similarly, the double-bit error count tracks the number of double-bit errors that have occurred since last clearing the status register (the double-bit error count stops at $17_8$). Any DCA:5 function clears the a or b status register, depending on which channel is active. A DCA:0 function will also clear the status register.

### DCA1:13 - Read Status 1

This function transmits one of two parcels to the accumulator. This function uses modifier bit M0.

Refer to Figure 5-11 for the accumulator format. If accumulator bit $2^0$ is 0, this function returns a general drive status or a selected drive status. The even channel returns the a status and the odd channel returns the b status (data from channels a and b). These status parcels vary with the type of disk storage unit; refer to the appropriate disk storage unit documentation for more information.

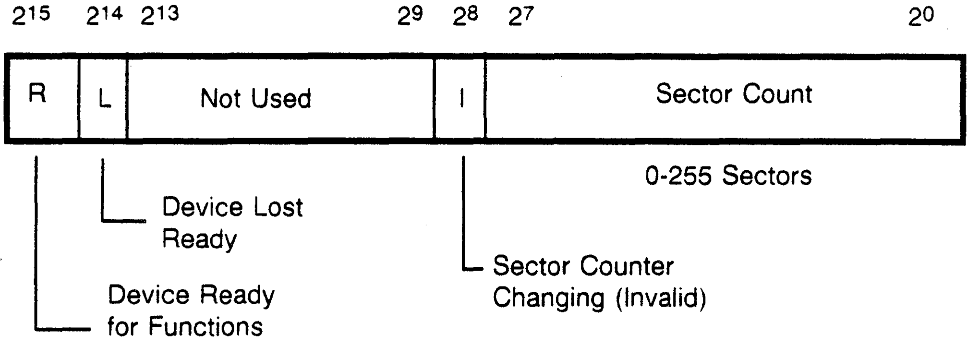


Figure 5-11. DCA1:13 (Accumulator = 1) Status Parcel

If accumulator bit $2^0$ is 1, this function returns the sector count, the sector count status, and the drive ready status. If bit $2^8$ is set, the sector counter is changing and the count is invalid. Bit $2^{15}$ is the current state of the ready line. Bit $2^{14}$ indicates that the ready line dropped momentarily. Bit $2^{14}$ is cleared by function DCA1:0.

### DCA1:16 - Write Disk Channel Parameter

The DCA:16 function transfers the accumulator contents to the parameter register. This function sets up parameter values for a subsequent DCA:5 function. This function uses modifier bit M0.

### DCA1:17 - Write Channel Adapter Modes

This function selects operating and diagnostic modes for the channel using accumulator bits. Different modes are selected depending on the even or odd value of the channel number in the instruction. Refer to Table 5-6 for the accumulator bit assignments. This function uses modifier bit M0.

**Note:** For the even channel, if the send resume count is not defined (bits $2^1$, $2^2$, and $2^3$ are cleared) the send resume count defaults to 12.

Table 5-6. DCA1:17 Accumulator Bit Assignments

| Accumulator Bits | "a" Even Channels | "b" Odd Channels |
|---|---|---|
| $2^0$ | 50 Mbytes/s (input cable) | 50 Mbytes/s (output cable) |
| $2^1$ | Send resume (count = 1) | Allow continuous data |
| $2^2$ | Send resume (count = 8) | Not used |
| $2^3$ | Send resume (count = 16) | Not used |
| $2^4$ | Toggle input data parity bit | Toggle output data parity bit |
| $2^5$ | Toggle status parity bit | Toggle parameter parity bit |
| $2^6$ | Toggle control parity bit | Toggle code parity bit |
| $2^7$ | Loop-back mode | Not used |
| $2^8$ through $2^{15}$ | Not used | Not used |

# DCA-2 CHANNEL ADAPTER

The DCA-2 disk channel adapter enables the IOS-E to communicate with disk drives. Specific disk drives associated with the DCA-2 include the DD-60, and DD-61. Each DCA-2 can control up to eight daisy-chained DD-60s, or eight daisy-chained DD-61s.

## Data Transfers

The DCA-2 transfers data to or from an external device at a rate of 3 Mbytes/s to 50 Mbytes/s. The DCA-2 operates in a half-duplex mode; that is, data transfers can take place on either channel in only one direction at a time. Data between the mainframe and the DCA-2 is SECDED protected; data from the DCA-2 to the external device is parity protected. Each data transfer between the channel adapter and peripheral device is 1 to 2,048 words long; each word is 64 bits.

## Channel Signals

Refer to Figure 5-12 for an illustration of the DCA-2 channel signals. The DCA-2 channel signals and channel signal definitions are provided in the following list.

Signal — Definition

Select Out — The Select Out signal is sent from the channel adapter to the disk drive. This signal selects a drive and maintains the selection. When this line is inactive, the drive releases the Bus A line.

Slave In — The Slave In signal is sent by the disk drive to the channel adapter. This signal acknowledges channel adapter control sequences, terminates data transfers, and acknowledges request sequences.

Master Out — The Master Out signal is sent from the channel adapter to the disk drive. This signal initiates or terminates data transfers, request interrupts, request drive interrupts, request transfer settings, and resets disk drives.

Sync In — The Sync In signal is sent from the disk drive to the channel adapter. This signal acknowledges the bus control byte during a bus control sequence. The Sync In signal also indicates when valid read data is on the bus.

Sync Out — The Sync Out signal is sent from the channel adapter to the disk drive. This signal acknowledges the bus control byte during a bus control sequence. The Sync In signal also indicates when valid write data is on the bus.

Bus A — The channel adapter uses Bus A for data transfers and control sequences. Bus A consists of eight data lines and one parity line.

**Signal** — **Definition**

Bus B — The channel adapter uses Bus B for data transfers and control sequences. Bus B consists of eight data lines and one parity line.

**Note:** Bus A and Bus B use odd parity.

Attention In — The Attention In signal is sent from the external device to the channel adapter when a command is complete.

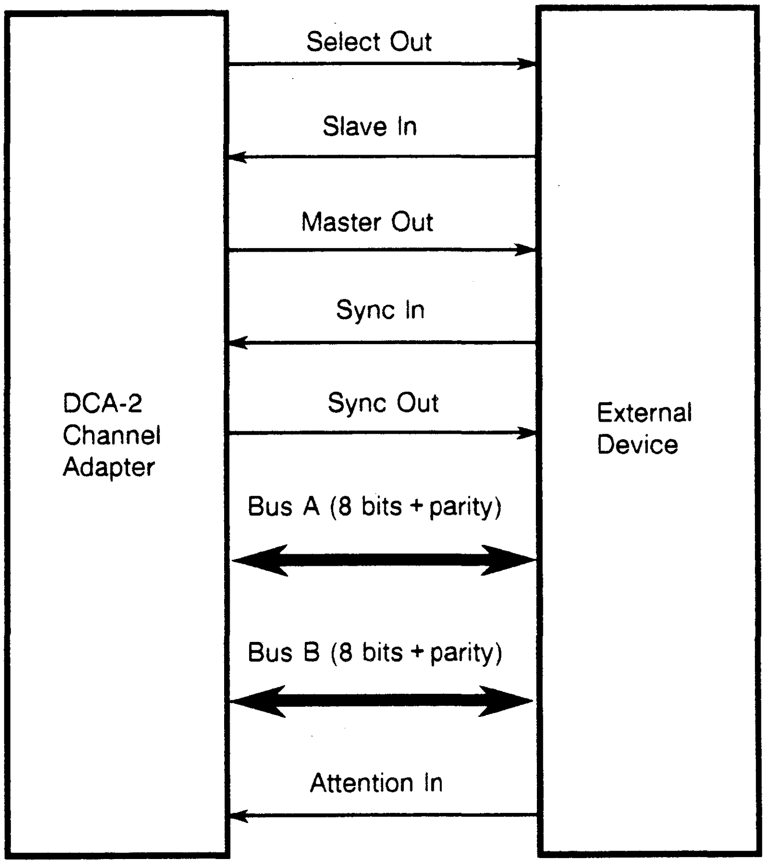


Figure 5-12. DCA-2 Channel Signals

## Channel Functions

Some of the functions have different uses depending on whether the function is issued to an even-numbered or an odd-numbered channel. Modifier bits M0 through M3 affect some of the functions. Table 5-7 lists the functions and the ways each function can be used.

Table 5-7. DCA-2 Disk Channel Functions

| Function | Description |
|---|---|
| DCA2:0 | Clear channel<br>M3, M2, M1 = 111: channel adapter master clear |
| DCA2:1 | Start transfer<br>Start input transfer (even channel)<br>M2, M1, M0 = 001: read status data<br>M2, M1, M0 = 011: read sequencer program<br>M2, M1, M0 = 101: read ECC registers<br>Start output transfer (odd channel)<br>M1, M0 = 00: write control data<br>M1, M0 = 10: write sequencer program |
| DCA2:2 | Write ID parameter 0 (cylinder address) |
| DCA2:3 | Write ID parameter 1 (head and sector address) |
| DCA2:4 | Write transfer count |
| DCA2:5 | Write starting sequencer address and I register |
| DCA2:6 | Clear interrupt enable flag |
| DCA2:7 | Set interrupt enable flag |
| DCA2:10 | Read local memory address |
| DCA2:11 | Read inverted parcel count |
| DCA2:12 | Read status 0 |
| DCA2:13 | Read status 1 |
| DCA2:14 | Write local memory address |
| DCA2:15 | Write parcel count |
| DCA2:16 | Write control registers j and k |
| DCA2:17 | Even channel: write mode select<br>Odd channel: write bit stream number |

### DCA2:0 - Clear Channel

This function aborts any transfer in progress and clears the channel busy and done flags. If modifier bits M1, M2, and M3 are set, this function also performs a master clear of the channel adapter.

### DCA2:1 Start Transfer

This function initiates a data transfer from the channel adapter to EIOP local memory, or a data transfer from EIOP local memory to the channel adapter. Refer to the following subsections for information on the DCA2:1 input and output transfer functions.

### DCA2:1 - Start Input Transfer (Even Channel)

This function initiates a data transfer from the channel adapter to EIOP local memory. Functions DCA2:14 and DCA2:15 must be issued before this function to set up the local memory address and parcel count. Function DCA2:0 must be issued after this function to clear the channel adapter control logic and buffered data. Modifier bits M1 through M3 must be set for function DCA2:0.

Modifier bits M0 through M2 affect the operation of this function. Bit M0 must be 1; this causes the channel adapter to use the busy, done, and interrupt flags for the odd channel while the IOP control logic uses the flags for the even channel.

The operation of this function for the three defined combinations of M0 through M3 is shown below.

| M3 | M2 | M1 | M0 | Operation |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | Read status data. This function is used in conjunction with a device command. It works similarly to function DCA2:5, except that data is transferred to local memory. When this function issues, the accumulator must contain the sequencer starting address and a value for the *i* register (the *i* register loads commands sent to a drive). Refer to function DCA2:5 for the accumulator format. |
| 0 | 0 | 1 | 1 | Read sequencer program. This function transfers the contents of the sequencer RAM (1,024 words × 32 bits) from the channel adapter to local memory. Bits $2^{16}$ through $2^{31}$ of a word are transferred first, followed by bits $2^{0}$ through $2^{15}$. |
| 0 | 1 | 0 | 1 | Read error correction code (ECC) registers. This function transfers the contents of eight 4-byte ECC registers to local memory. |

### DCA2:1 - Start Output Transfer (Odd Channel)

This function initiates a data transfer to the channel adapter from IOP local memory. Functions DCA2:14 and DCA2:15 must be issued before and after this function to set up the local memory address and parcel count. Function DCA2:0 must be issued after this function to clear the channel adapter control logic and buffered data. Modifier bits M1 through M3 must be set for function DCA2:0.

Modifier bits M0 and M1 affect the operation of this function. Bit M0 must be 0; this causes the channel adapter to use the busy, done, and interrupt flags for the even channel while the IOP DMA control logic uses the flags for the odd channel. Using both sets of flags allows the interface to be monitored separately from the DMA channel.

The operation of this function for both defined combinations of M0 through M3 is shown below:

| M3 | M2 | M1 | M0 | Operation |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Write control data. This function is used in conjunction with a device command. It works similarly to function DCA2:5, except that data is transferred from local memory. When this function issues, the accumulator must contain the sequencer starting address and a value for the *i* register; refer to function DCA2:5 for the accumulator format. |
| 0 | 0 | 1 | 0 | Write sequencer program. This function transfers local memory data to the sequencer RAM (1,024 words × 32 bits). Bits $2^{16}$ through $2^{31}$ of a word are transferred first, followed by bits $2^{0}$ through $2^{15}$. |

### DCA2:2 - Write ID Parameter 0

This function transfers the cylinder address (ID parameter 0) from the accumulator to the cylinder address register. (ID parameter 0 is bytes 2 and 3 of the sector ID field.) Refer to Figure 5-13 for the DCA2:2 Parameter 0 format.

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C 15 | C 14 | C 13 | C 12 | C 11 | C 10 | C 9 | C 8 | C 7 | C 6 | C 5 | C 4 | C 3 | C 2 | C 1 | C 0 |

C = Cylinder Address (0-$5062_8$)

Figure 5-13. DCA2:2 Parameter 0

### DCA2:3 - Write ID Parameter 1

This function transfers the accumulator contents to ID parameter 1: the sector address, head group number, and 3 control bits. (ID parameter 1 is bytes 4 and 5 of the sector ID field.) Refer to Figure 5-14 for the DCA2:3 Parameter 1 format.

### DCA2:4 - Write Transfer Count

This function uses the accumulator contents as the parcel count for a data transfer. The transfer count for a DD-60 disk storage unit is $20154_8$ (8,300) parcels. Refer to Table 5-8 for the octal and decimal transfer counts for the DCA2:4 function.

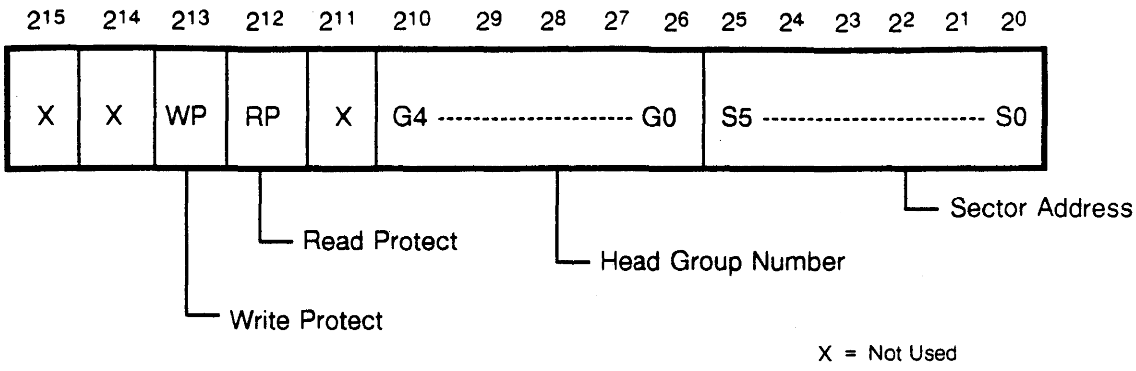


Figure 5-14. DCA2:3 Parameter 1

Table 5-8. DCA2:4 Function Transfer Counts

| Parcel | Octal | Decimal |
|---|---|---|
| Data parcels | 20000 | 8192 |
| ID parcels | 24 | 20 |
| ECC parcels | 20 | 16 |
| Defect parcels | 110 | 72 |

### DCA2:5 - Write Starting Sequencer Address and i Register

This function uses the accumulator contents as two fields for the channel adapter. Bits $2^0$ through $2^6$ are the sequence starting address. Bits $2^8$ through $2^{15}$ are transmitted to the i register, which is used for a device address or a bus control code depending on the instruction sequence. (The device address specifies which device the data is going to, and the bus control code specifies which operation is taking place.) Bit $2^7$ is not used. Refer to Figure 5-15 for the DCA2:5 control parcel.

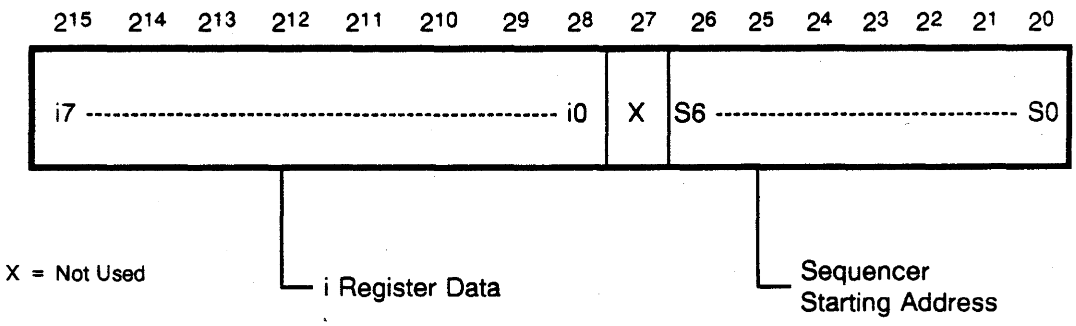

2^15 2^14 2^13 2^12 2^11 2^10 2^9 2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
i7 ---------------------------------------------------- i0
X
S6 -------------------------------- S0
X = Not Used
i Register Data
Sequencer Starting Address

Figure 5-15. DCA2:5 Control Parcel

### DCA2:6 - Clear Interrupt Enable Flag

This function clears the channel interrupt enable flag.

### DCA2:7 - Set Interrupt Enable Flag

This function sets the channel interrupt enable flag.

### DCA2:10 - Read Local Memory Address

This function transfers the contents of the EIOP local memory address register to the accumulator.

### DCA2:11 - Read Inverted Parcel Count

This function transfers the inverted contents of the parcel count register to the accumulator. To obtain the correct parcel count, subtract the accumulator contents from $177777_8$.

### DCA2:12 - Read Status 0

This function transmits 1 of 4 status parcels to the accumulator. The contents of accumulator bits $2^0$ and $2^1$ when the function issues determines which status parcel is returned. Refer to Table 5-9 which details the 4 status parcels.

The even and odd channels return separate status parcels, except for the input tags (Attn-In, Slave-In, and Sync-In). Each successive DCA2:12 function returns a new set of input tags.

### DCA2:13 - Read Status 1

This function transmits 1 of 4 status parcels to the accumulator. The contents of accumulator bits $2^0$ and $2^1$, when the function issues, determines which status parcel is returned. Refer to Table 5-10 for descriptions of the four status parcels.

The even and odd channels return separate status parcels for the ending status, initial status, and ID parameter 1 (head and sector). The even and odd channels return a common status parcel for ID parameter 0 (cylinder).

### DCA2:14 - Write Starting Address

This function transfers the contents of the accumulator to the local memory address register.

### DCA2:15 - Write Parcel Count

This function transfers the contents of the accumulator to the parcel count register.

Table 5-9. DCA2:12 Status Parcels

| Bit | Accumulator Bits $2^1$ and $2^0$ | | | |
|---|---|---|---|---|
| | Bits = 00 (binary) | Bits = 01 (binary) | Bits = 10 (binary) | Bits = 11 (binary) |
| $2^0$ | Channel adapter status 0 | Syndrome bit 0 | Transfer count $2^0$ | Attn-in |
| $2^1$ | Channel adapter status 1 | Syndrome bit 1 | Transfer count $2^1$ | Attn-out |
| $2^2$ | Channel adapter status 2 | Syndrome bit 2 | Transfer count $2^2$ | Sync-in |
| $2^3$ | Channel adapter status 3 | Syndrome bit 3 | Transfer count $2^3$ | Not used |
| $2^4$ | Channel adapter status 4 | Syndrome bit 4 | Transfer count $2^4$ | Select out |
| $2^5$ | Channel adapter status 5 | Syndrome bit 5 | Transfer count $2^5$ | Master out |
| $2^6$ | Channel adapter status 6 | Syndrome bit 6 | Transfer count $2^6$ | Sync-out |
| $2^7$ | Channel adapter status 7 | Syndrome bit 7 | Transfer count $2^7$ | Not used |
| $2^8$ | Single-bit SECDED error | Single-bit error count $2^0$ | Transfer count $2^8$ | Not used |
| $2^9$ | Double-bit SECDED error | Single-bit error count $2^1$ | Transfer count $2^9$ | Not used |
| $2^{10}$ | Voltage fault -4.5 V | Single-bit error count $2^2$ | Transfer count $2^{10}$ | Not used |
| $2^{11}$ | Voltage fault +5.0 V | Single-bit error count $2^3$ | Transfer count $2^{11}$ | Not used |
| $2^{12}$ | Sequencer parity error 0 | Double-bit error count $2^0$ | Transfer count $2^{12}$ | Not used |
| $2^{13}$ | Sequencer parity error 1 | Double-bit error count $2^1$ | Transfer count $2^{13}$ | Not used |
| $2^{14}$ | Sequencer parity error 2 | Double-bit error count $2^2$ | Transfer count $2^{14}$ | Not used |
| $2^{15}$ | Sequencer parity error 3 | Double-bit error count $2^3$ † | Transfer count $2^{15}$ | Not used |

† The double-bit error count stops at $17_8$.

### DCA2:16 - Write Control Registers j and k

This function uses the accumulator contents as two fields for the channel adapter. Bits $2^0$ through $2^7$ are transmitted to the j register. Bits $2^8$ through $2^{15}$ are transmitted to the k register. Refer to Figure 5-16 for the DCA2:16 control register bits format.

The sequencer stores bus control commands in the j and k registers; they are used for bus A output and bus control codes or for bus A or bus B information octets.

Table 5-10. DCA2:13 Status Parcels

| Bit | Accumulator Bits $2^1$ and $2^0$ | | | |
|---|---|---|---|---|
| | Bits = 00 (binary) | Bits = 01 (binary) | Bits = 10 (binary) | Bits = 11 (binary) |
| $2^0$ | Ending status 0 | ID parameter 0 bit 0 | ID parameter 1 bit 0 | Not used |
| $2^1$ | Ending status 1 | ID parameter 0 bit 1 | ID parameter 1 bit 1 | Not used |
| $2^2$ | Ending status 2 | ID parameter 0 bit 2 | ID parameter 1 bit 2 | Not used |
| $2^3$ | Ending status 3 | ID parameter 0 bit 3 | ID parameter 1 bit 3 | Not used |
| $2^4$ | Ending status 4 | ID parameter 0 bit 4 | ID parameter 1 bit 4 | Not used |
| $2^5$ | Ending status 5 | ID parameter 0 bit 5 | ID parameter 1 bit 5 | Not used |
| $2^6$ | Ending status 6 | ID parameter 0 bit 6 | ID parameter 1 bit 6 | Not used |
| $2^7$ | Ending status 7 | ID parameter 0 bit 7 | ID parameter 1 bit 7 | Not used |
| $2^8$ | Initial status 0 | ID parameter 0 bit 8 | ID parameter 1 bit 8 | Not used |
| $2^9$ | Initial status 1 | ID parameter 0 bit 9 | ID parameter 1 bit 9 | Not used |
| $2^{10}$ | Initial status 2 | ID parameter 0 bit 10 | ID parameter 1 bit 10 | Not used |
| $2^{11}$ | Initial status 3 | ID parameter 0 bit 11 | ID parameter 1 bit 11 | Not used |
| $2^{12}$ | Initial status 4 | ID parameter 0 bit 12 | ID parameter 1 bit 12 | Not used |
| $2^{13}$ | Initial status 5 | ID parameter 0 bit 13 | ID parameter 1 bit 13 | Not used |
| $2^{14}$ | Initial status 6 | ID parameter 0 bit 14 | ID parameter 1 bit 14 | Not used |
| $2^{15}$ | Initial status 7 | ID parameter 0 bit 15 | ID parameter 1 bit 15 | Not used |

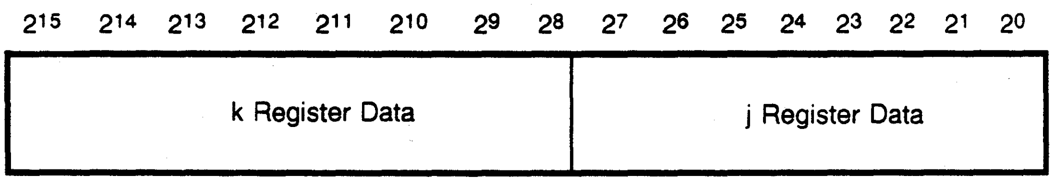


Figure 5-16. DCA2:16 Control Register Bits

### DCA2:17 - Write Mode Select

This function transmits the lower byte of the accumulator to the mode select register, or the lower byte of the accumulator to the bit stream register in the sequencer. Refer to the following subsections for information on the DCA2:17 write mode select for the even and odd channels.

### DCA2:17 - Write Mode Select (Even Channel)

This function transmits bits $2^0$ through $2^7$ of the accumulator to the mode select register in the sequencer. Bits $2^8$ through $2^{15}$ are not used. The 8 mode bits can be altered for any function sequence. Refer to Figure 5-17 for the DCA2:17 (even channel) write mode select bits.

<table>
<tr><th>$2^{15}$</th><th>$2^{14}$</th><th>$2^{13}$</th><th>$2^{12}$</th><th>$2^{11}$</th><th>$2^{10}$</th><th>$2^9$</th><th>$2^8$</th><th>$2^7$</th><th>$2^6$</th><th>$2^5$</th><th>$2^4$</th><th>$2^3$</th><th>$2^2$</th><th>$2^1$</th><th>$2^0$</th></tr>
<tr><td colspan="8">Not Used</td><td colspan="8">Mode Select</td></tr>
</table>

Figure 5-17. DCA2:17 (Even Channel) Write Mode Select Bits

### DCA2:17 - Write Mode Select (Odd Channel)

This function transmits bits $2^0$ through $2^7$ of the accumulator to the bit stream number register. Bits $2^9$ through $2^{15}$ are not used. Refer to Figure 5-18 for the DCA2:17 (odd channel) bit stream select bits.

A bit that is set causes that bit of the A bus and B bus to be replaced with a bit stream calculated by summing the parity stream data with the other seven data streams. This function can be used after a head, head preamp, read circuit, or write circuit failure within the disk drive. Refer to the appropriate disk manual for information on these failures.

<table>
<tr><th>$2^{15}$</th><th>$2^{14}$</th><th>$2^{13}$</th><th>$2^{12}$</th><th>$2^{11}$</th><th>$2^{10}$</th><th>$2^9$</th><th>$2^8$</th><th>$2^7$</th><th>$2^6$</th><th>$2^5$</th><th>$2^4$</th><th>$2^3$</th><th>$2^2$</th><th>$2^1$</th><th>$2^0$</th></tr>
<tr><td colspan="8">Not Used</td><td colspan="8">Bit Stream Select</td></tr>
</table>

Figure 5-18. DCA2:17 (Odd Channel) Bit Stream Select Bits

# HCA-3 AND HCA-4 CHANNEL ADAPTERS

The HCA-3 and HCA-4 channel adapters are the IOS-E implementation of the High Performance Parallel Interface (HIPPI), a preliminary ANSI-standard communication channel optimized for large block transfers. The HCA-3 is the input channel, and the HCA-4 is the output channel.

HIPPI provides high-speed data transfers over long distances, using either a 32-bit or 64-bit data bus. With a 32-bit data bus, the peak transfer rate is 100 Mbytes/s; a single 50-pair cable (cable A) contains all the data and control lines. Increasing the bus width to 64 bits doubles the peak transfer rate to 200 Mbytes/s but requires the addition of a second 50-pair cable (cable B).

The HIPPI specification limits the maximum distance between two devices to 82 ft (25 m). However, the distance can be greatly increased by using a passive (nonbuffering) extender, such as a fiber-optic link. A look-ahead control protocol minimizes the effect of propagation delays and allows sustained transfer rates to approach the peak transfer rates at distances up to 37 miles (60 km).

## Definitions

The following definitions are used in the HIPPI protocol description.

Word — Definition

Source — The transmitting device on a HIPPI channel; in the IOS-E, an HCA-4 channel adapter.

Destination — The receiving device on an HIPPI channel; in the IOS-E, an HCA-3 channel adapter.

Word — A 64-bit unit of information. In a HIPPI channel operating at 200 Mbytes/s (64-bit data bus), each burst consists of from 1 to 256 words.

**Note:** This definition is provided for consistency with Cray Research conventions. It does not agree with the definition provided in the HIPPI specification.

Halfword — A 32-bit unit of information. In a HIPPI channel operating at 100 Mbytes/s (32-bit data bus), each burst consists of from 1 to 256 halfwords.

**Note:** This definition is provided for consistency with Cray Research conventions. It is not part of the HIPPI specification.

Packet — A data set sent from a source to a destination. Each packet consists of one or more bursts.

**Note:** This packet is not the same as, and should not be confused with UNICOS I/O request packets sent to the IOS-E.

Word Definition

Burst
A group of from 1 to 256 halfwords (100-Mbyte/s mode) or from 1 to 256 words (200-Mbyte/s mode). Each packet contains one or more bursts.

Short burst
A burst containing less than 256 halfwords or words. There can be only one short burst in each packet; the short burst must be either the first or last burst of the packet.

Connection
The condition of a HIPPI channel when data transfers from source to destination are possible.

I field
A 32-bit information field that is sent from a source to a destination when a connection is established. The contents of the I field are not defined by the HIPPI specification.

**Note:** In the IOS-E implementation, the I field is 64 bits wide in 200-Mbyte/s mode; the upper 32 bits are used only for diagnostic purposes.

LLRC
Length/longitudinal redundancy checkword (LLRC), is a halfword or a word transmitted on the data bus immediately after each burst. The LLRC provides a data integrity check and verifies that the number of halfword/words received is the same as the number of halfwords/words transmitted.

### HIPPI Signals

Refer to Figure 5-19 for an illustration of the HIPPI channel signals. Each signal is defined in the following list:

Signal Definition

Request
The source sets the Request signal to indicate a connection is requested; the Request signal remains set for the duration of the connection. The Request signal clears when the source drops the connection or when the Connect signal clears.

Connect
The destination sets the Connect signal in response to a Request signal to indicate that the connection has been made. The destination clears the Connect signal to drop the connection. The destination can also set the Connect signal momentarily, then clear it, to reject the connection.

Data
The data bus consists of either 32 or 64 bits.

Parity
There is an odd parity bit for each byte on the data bus. Therefore, there are either 4 or 8 parity bits.

Signal Definition

Ready The destination sets the Ready signal one time for each burst it can accept from the source. Using a look-ahead feature, the destination can send multiple (up to 63) Ready signals to indicate that it can receive multiple bursts.

Packet The source sets the Packet signal before the beginning of each burst and clears the Burst signal at the end of the burst.

Clock The source transmits a 25-MHz (40-ns clock period) symmetric signal.

Interconnect The source and destination transmit one or two Interconnect signals to each other to indicate that they are connected and powered up. For 100-Mbyte/s operation, only one signal (interconnect A) in each direction is used. For 200-Mbyte/s operation, two signals (interconnect A and interconnect B) are used in each direction.

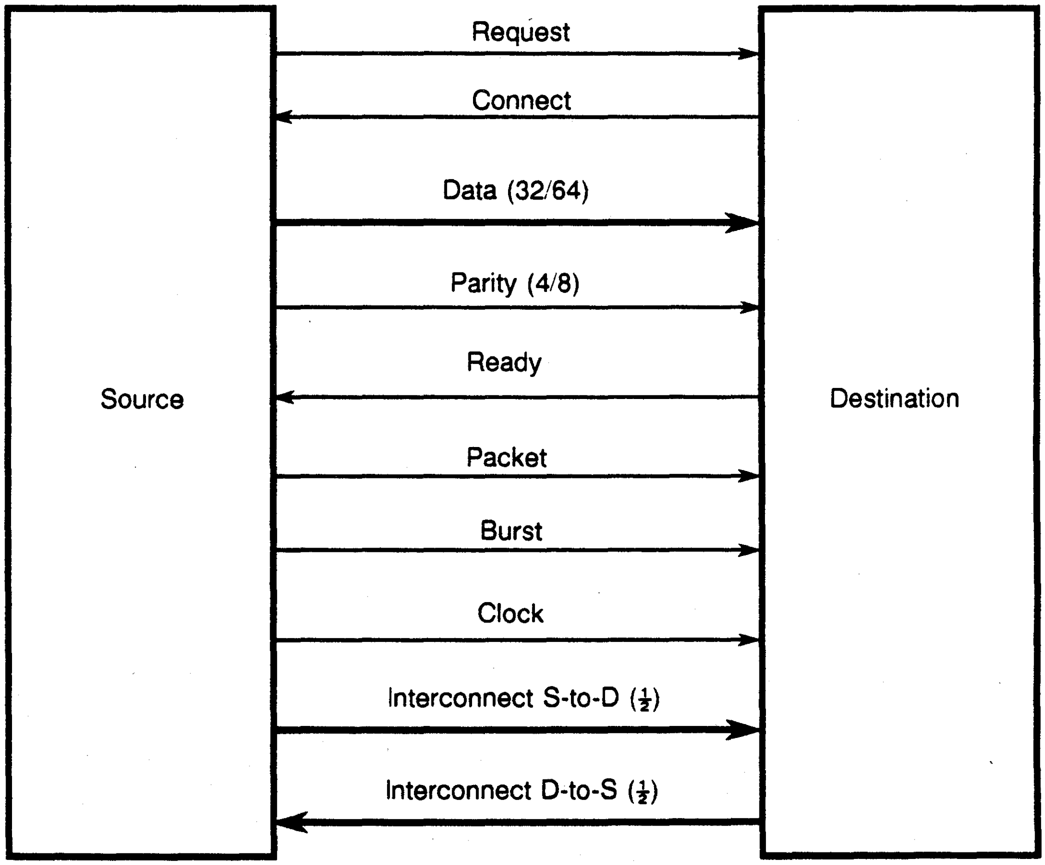


Figure 5-19. HIPPI Signals

### HIPPI Protocol

A typical data transfer uses these signals as follows:

1. The source places the I field on the data bus and sets the Request signal to initiate a connection.
2. The destination reads the I field from the data bus and sets the connect signal to complete the connection.
3. The destination sets the Ready signal once for each burst it can receive. Throughout the transfer, the destination repeatedly sets Ready.
4. The source sets the Packet signal to indicate the beginning of a packet.
5. The source sets the Burst signal to indicate the beginning of a burst.
6. The source transfers a burst of data to the destination at a rate of 1 halfword or word per clock period. Delays are not permitted during the burst.
7. The source clears the Burst signal to indicate the end of the burst.
8. The source sends the LLRC to the destination on the clock period following the last halfword/word of the burst.
9. Steps 5 through 8 are repeated for each burst in the packet. Delays may occur between bursts.
10. The source clears the Packet signal to indicate the end of the packet.
11. Steps 4 through 10 are repeated for each packet.
12. The source clears the Request signal to terminate the connection.
13. The destination clears the Connect signal.

## HCA-3 - Channel Adapter Operation

The HCA-3 channel adapter is a HIPPI destination (input channel), transferring data from an external device to the I/O buffer. The HCA-3 can receive up to 32,768 words in a single channel activation. All activations are for an integral number of full bursts.

The HCA-3 can use either I/O buffer pointer a or pointer b for the buffer address during a transfer. Of the two IOP channels associated with an HCA-3, the even channel controls transfers using pointer a; the odd channel controls transfers using pointer b.

In the 100-Mbyte/s mode, the HCA-3 assembles two successive halfwords received from the channel into a single word for the I/O buffer. The first halfword corresponds to bits $2^{32}$ through $2^{63}$ of the buffer word; the second halfword corresponds to bits $2^{0}$ through $2^{31}$. If the HCA-3 receives a burst with an odd number of halfwords, it clears bits $2^{0}$ through $2^{31}$ of the final buffer word.

Several HCA-3 channel functions require the programmer to specify the pointer (channel number bit $2^0$) in modifier bit M0 (*d* field or B register bit $2^5$). For example, if an HCA-3 is connected to channels 30 and 31, pointer a is specified by issuing a channel function with *d* field/B register set to $30_8$; pointer b is specified by setting the *d* field/B register to $71_8$. Modifier bits M1 through M3 are not used by any HCA-3 channel function.

A feature called activation stacking allows successive channel transfers to proceed without a delay for the channel to be reprogrammed between transfers. To use activation stacking, the transfers must alternate between the two buffer pointers (for example, pointer a for the first activation, pointer b for the second, pointer a for the third, etc.). Activate the first transfer using pointer a, and then activate the second transfer using pointer b. The first transfer starts immediately, and the second transfer starts when the first transfer is completed. The third transfer can be initiated (using pointer a) while the second transfer is in progress and automatically starts when the second transfer is completed. This process can be continued until all data is transferred.

In addition to the busy and done flags used by all channels, each HCA-3 channel has an activation stacked flag which indicates a transfer waiting to start. If a channel is activated when the other channel is busy, the activation stacked flag sets until the transfer can begin. Then the activation stacked flag clears, the busy flag sets, and the transfer proceeds.

The HCA-3 generates an interrupt request when any of several conditions occurs. (The cause of the interrupt request can be determined by reading a status parcel. Refer to function HCA3:12). Two conditions are called data interrupts.

- The halfword count decrements to 0. (Refer to function HCA3:5). This is the only interrupt condition that sets the done flag.
- The source clears the Packet signal, indicating the end of a packet.

The HCA-3 can store up to 63 successive interrupt requests caused by the above conditions. Along with each interrupt request, the HCA-3 stores the packet status (reason for interrupt: halfword count equal to 0 or packet end) and the packet length. The packet status and the packet length are parity protected. Refer to function HCA3:12 for more read status and packet length information.

Other conditions are called nondata interrupts.

- The source requests or ends a connection.
- A burst is received that was not enabled for a Ready signal (buffer overrun).
- The incoming interconnect-A signal changes state.
- The incoming interconnect-B signal changes state (200-Mbyte/s mode only).

No packet status or packet length is saved for these conditions.

Refer to Table 5-11 for the HCA-3 channel functions.

Table 5-11. HCA-3 Channel Functions

| Function | Description |
|---|---|
| HCA3:0 | Clear channel |
| HCA3:4 | Accept/reject connection or reset channel |
| HCA3:5 | Write halfword count and activate channel |
| HCA3:6 | Clear interrupt enable flag |
| HCA3:7 | Set interrupt enable flag |
| HCA3:12 | Read status or packet length |
| HCA3:13 | Read I field or packet length |
| HCA3:17 | Write modes |

### HCA3:0 - Clear Channel

This function clears the channel busy, done, and activation stacked flags. This function uses modifier bit M0.

### HCA3:4 - Accept/Reject Connection or Reset Channel

Refer to Figure 5-20 for an illustration of the HCA3:4 control parcel. The function uses accumulator bits $2^0$ through $2^2$ to control operation of the channel. Bit $2^0$ causes the HCA-3 to accept a connection from the source. It sets the Connection signal. Bit $2^1$ causes the HCA-3 to reject a connection. It sets the Connection signal momentarily, and then clears it. Bit $2^2$ causes the HCA-3 to abort a connection. The following items are cleared:

- The Connect signal
- The busy, done, and all error flags for both channels
- All current and pending interrupt requests for both channels

This function uses modifier bit M0.

### HCA3:5 - Write Halfword Count and Activate Channel

This function uses the accumulator contents as the halfword count for the transfer. The channel is always activated to accept an integral number of bursts. Therefore, in 100-Mbyte/s mode, the halfword count is assumed to be a multiple of 256; accumulator bits $2^0$ through $2^7$ are ignored. In 200-Mbyte/s mode, accumulator bits $2^0$ through $2^8$ are ignored. An accumulator value of 0 causes a transfer of 65,536 halfwords (32,768 words). This function uses modifier bit M0.

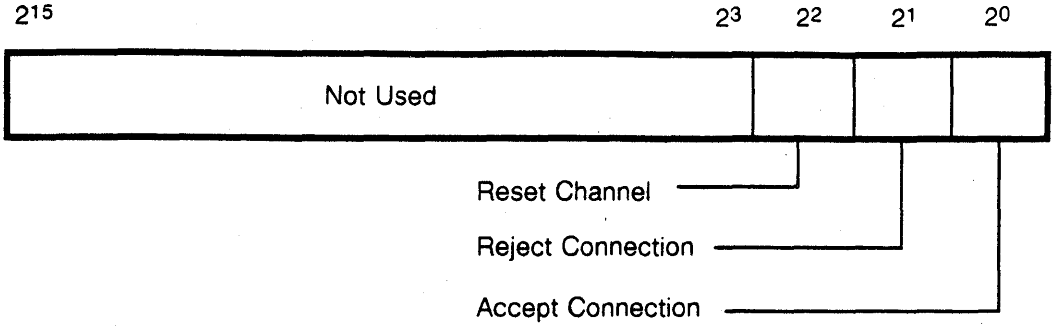


Figure 5-20. HCA3:4 Control Parcel

If no transfer using the other buffer pointer is in progress, this function starts a transfer immediately. If a transfer using the other pointer is in progress, the transfer is delayed (stacked) until the other transfer completes.

The HCA-3 uses activation stacking. This feature allows the channel to continuously receive data.

### HCA3:6 - Clear Interrupt Enable Flag

This function clears the channel interrupt enable flag. This function uses modifier bit M0.

### HCA3:7 - Set Interrupt Enable Flag

This function sets the channel interrupt enable flag. This function uses modifier bit M0.

### HCA3:12 - Read Status or Packet Length

This function transmits one of three status parcels to the accumulator. The contents of accumulator bits $2^0$ and $2^1$ when the function issues determines which status parcel is returned. Refer to Table 5-12 for descriptions of the three parcels. Modifier bit M0 is not used.

If accumulator bits $2^0$ and $2^1$ are 0's, this function transmits 15 status bits to the accumulator. Some of these bits indicate the cause of the most recent interrupt request, while other bits provide general status and error information.

If accumulator bit $2^0$ is 1 and bit $2^1$ is 0, this function transfers the packet length (in halfwords) to the accumulator. If the interrupt was caused by a packet end, this function returns the actual packet length. If the interrupt was caused by the transfer length decrementing to 0, HCA3:12 returns the number of halfwords received since the beginning of the packet.

**Note:** The packet length must be read once each time the channel status parcel indicates an interrupt request caused by the halfword count decrementing to 0 or by a packet end. The packet length must not be read at any other time.

Table 5-12. HCA3:12 Status Parcels

| Bit | Accumulator Bits $2^1$ and $2^0$ | | |
|---|---|---|---|
| | Bits = 00 (binary) | Bits = 01(binary) | Bits = 10 or 11(binary) |
| $2^0$ | Connection requested interrupt | Packet length bit $2^0$ | Interconnect-A signal in |
| $2^1$ | Connection ended interrupt | Packet length bit $2^1$ | Interconnect-B signal in |
| $2^2$ | Half-word count decremented to 0 interrupt | Packet length bit $2^2$ | Request signal in |
| $2^3$ | Packet end interrupt | Packet length bit $2^3$ | Connect signal out |
| $2^4$ | Packet status parity error | Packet length bit $2^4$ | Not used |
| $2^5$ | Packet length parity error | Packet length bit $2^5$ | Not used |
| $2^6$ | Data parity error, or LLRC parity error | Packet length bit $2^6$ | Not used |
| $2^7$ | LLRC error | Packet length bit $2^7$ | Not used |
| $2^8$ | Initial short burst | Packet length bit $2^8$ | Not used |
| $2^9$ | Buffer overrun interrupt | Packet length bit $2^9$ | Not used |
| $2^{10}$ | Sequence error † | Packet length bit $2^{10}$ | Not used |
| $2^{11}$ | First burst of packet received interrupt ‡ | Packet length bit $2^{11}$ | Not used |
| $2^{12}$ | Interconnect transition interrupt | Packet length bit $2^{12}$ | Not used |
| $2^{13}$ | Not used | Packet length bit $2^{13}$ | Not used |
| $2^{14}$ | Illegal burst length | Packet length bit $2^{14}$ | Not used |
| $2^{15}$ | Activation stacked | Packet length bit $2^{15}$ | Not used |

† This indicates that the Request signal is dropping during a packet or a Packet signal is dropping during a burst.
‡ Refer to function HCA3:17.

If accumulator bit $2^1$ is a 1, this function transmits 4 status bits to the accumulator. These bits indicate the state of the Interconnect, Request, and Connect signals.

## HCA3:13 - Read I Field or Packet Length

This function transfers 16 bits of the LLRC error code or I field to the accumulator. (The LLRC error code is the exclusive OR of the expected and actual LLRCs.) Accumulator bits $2^0$ through $2^2$ determine which bits are transferred. Table 5-13 shows which bits are read. Modifier bit M0 is not used.

Table 5-13. HCA3:13 Status Parcel

| Accumulator Bits $2^2$ through $2^0$ (binary) | Result |
|---|---|
| 000 | LLRC error code bits $2^0$ through $2^{15}$ |
| 001 | LLRC error code bits $2^{16}$ through $2^{31}$ |
| 010 | LLRC error code bits $2^{32}$ through $2^{47}$ |
| 011 | LLRC error code bits $2^{48}$ through $2^{63}$ |
| 100 | I field bits $2^0$ through $2^{15}$ |
| 101 | I field bits $2^{16}$ through $2^{31}$ |
| 110 | I field bits $2^{32}$ through $2^{47}$ |
| 111 | I field bits $2^{48}$ through $2^{63}$ |

## HCA3:17 - Write Modes

This function uses accumulator bits $2^0$ through $2^3$ to select four diagnostic modes. Refer to Figure 5-21 for an illustration of the HCA3:17 control parcel.

Bit $2^0$ decrements the count of Ready signals that have been sent to the destination. This bit can be used to force buffer overrun errors.

Bit $2^1$ causes the parity bit stored with each packet status to be inverted before it is stored. When the packet status is read, a parity error occurs.

Bit $2^2$ causes the incoming interconnect-B signal to be treated as false, regardless of its actual state. The outgoing interconnect-B signal is not affected. This function must not be used during a connection.

Bit $2^3$ generates an interrupt request when the first burst of a packet is received and stored in the I/O buffer. This interrupt request is cleared by reading the status register (function HCA:12 with accumulator equal to 0). If a data interrupt occurs after a first-burst interrupt but before the status register is read, the data interrupt replaces the first-burst interrupt.

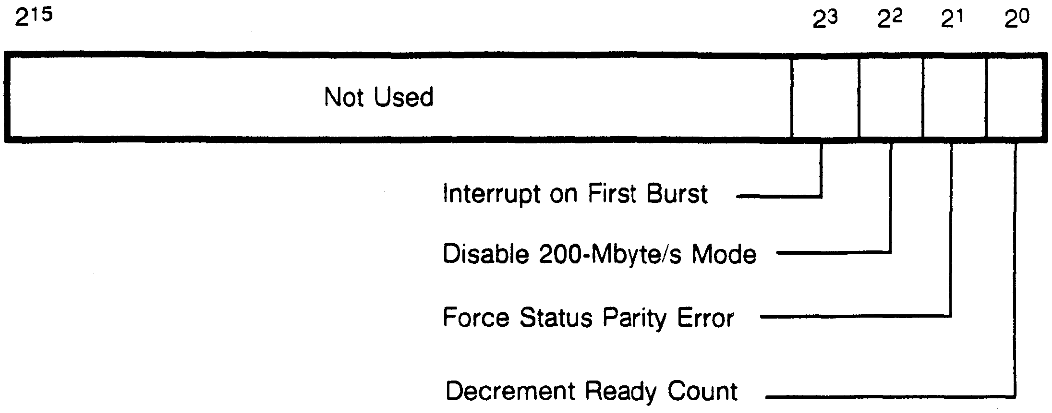


Figure 5-21. HCA3:17 Control Parcel

## HCA-4 - Channel Adapter Operation

The HCA-4 channel adapter is an HIPPI source (output channel), transferring data from the I/O buffer to an external device. The HCA-4 can transmit up to 32,768 words in a single channel activation.

The HCA-4 can use either I/O buffer pointer a or pointer b for the buffer address during a transfer. Of the two IOP channels associated with an HCA-4, the even channel controls transfers using pointer a; the odd channel controls transfers using pointer b.

In 100-Mbyte/s mode, the HCA-4 disassembles each word from the I/O buffer into two halfwords for the channel. The first halfword corresponds to bits $2^{32}$ through $2^{63}$ of the buffer word; the second halfword corresponds to bits $2^0$ through $2^{31}$. If the HCA-4 transmits a packet with an odd number of halfwords, it ignores bits $2^0$ through $2^{31}$ of the final buffer word.

Several HCA-4 channel functions require the programmer to specify the pointer (channel number bit $2^0$) in modifier bit M0 (*d* field or B register bit $2^5$). For example, if an HCA-4 is connected to channels 30 and 31, pointer a is specified by issuing a channel function with *d* field/B register set to $30_8$; pointer B is specified by setting the *d* field/B register to $71_8$. Modifier bits M1 through M3 are not used by any HCA-4 channel function.

A feature called activation stacking allows successive channel transfers to proceed without a delay for the channel to be reprogrammed between transfers. To use activation stacking, the transfers must alternate between the two buffer pointers (for example, pointer a for the first activation, pointer b for the second, pointer a for the third, etc.). Activate the first transfer using pointer a, then activate the second transfer using pointer b. The first transfer starts immediately, and the second transfer starts when the first transfer is completed. The third transfer can be initiated (using pointer a) while the second transfer is in progress and automatically starts when the second transfer completes. This process can be continued until all data is transferred.

In addition to the busy and done flags used by all channels, each HCA-4 channel has an activation stacked flag which indicates a transfer waiting to start. If a channel is activated when the other channel is busy, the activation stacked flag sets until the transfer can begin. Then the activation stacked flag clears, the busy flag sets, and the transfer proceeds.

Normally, each channel activation is treated as a separate HIPPI packet; the Packet signal sets at the beginning of the activation and clears at the end of the activation, even if another activation is stacked. However, multiple activations can be treated as a single packet by forcing the Packet signal to remain set between activations.

If the Packet signal is forced to remain set between activations, it must be cleared after the final activation of the packet. There are two ways to do this:

- Wait until all activations have completed, then force the Packet signal to clear.
- Restore normal operation of the Packet signal before initiating the final activation. When the final activation completes, the Packet signal automatically clears.

The HCA-4 generates an interrupt request when any of several conditions occurs:

- The halfword count decrements to 0. This condition indicates a completed activation and sets the done flag.
- The destination accepts, rejects, or aborts a connection.
- The destination clears the Connect signal when the Request signal is not active (connection ended condition).
- The incoming interconnect-A signal changes state.
- The incoming interconnect-B signal changes state (200-Mbyte/s mode only).

Refer to Table 5-14 for the HCA-4 channel functions.

Table 5-14. HCA-4 Channel Functions

| Function | Description |
|---|---|
| HCA4:0 | Clear channel |
| HCA4:2 | Write lower I field |
| HCA4:3 | Write upper I field |
| HCA4:4 | Control channel operation |
| HCA4:5 | Write halfword count and activate channel |
| HCA4:6 | Clear interrupt enable flag |
| HCA4:7 | Set interrupt enable flag |
| HCA4:12 | Read halfword count or status |
| HCA4:13 | Read error code |
| HCA4:17 | Write diagnostics modes |

### HCA4:0 - Clear Channel

This function clears the channel busy, done, and activation stacked flags. Modifier bit M0 determines which channel is affected.

### HCA4:2 - Write Lower I Field

This function performs two different functions, depending on the operating mode (refer to function HCA4:17). Modifier bit M0 is not used for this function.

- In normal operation, this function transfers the accumulator contents to bits $2^0$ through $2^{15}$ of the I field.
- In LLRC diagnostic mode or forced parity mode, this function transfers the accumulator contents to bits $2^{32}$ through $2^{47}$ of the I field.

### HCA4:3 - Write Upper I Field

This function performs two different functions, depending on the operating mode (refer to function HCA4:17).

- In normal operation, it transfers the accumulator contents to bits $2^{16}$ through $2^{31}$ of the I field.
- In LLRC diagnostic mode or forced parity mode, it transfers the accumulator contents to bits $2^{48}$ through $2^{63}$ of the I field.

### HCA4:4 - Control Channel Operation

The function uses accumulator bits $2^0$ through $2^3$ and $2^8$ to control operation of the channel. Refer to Figure 5-22 for an illustration of the HCA4:4 control parcel.

Bit $2^0$ causes the HCA-4 to request a connection to the destination device. It sets the Request signal and waits for a response from the destination.

Bit $2^1$ causes the HCA-4 to terminate a connection. It clears the Request signal.

Bit $2^2$ causes the HCA-4 to reset the channel. The following items are cleared:

- The Request signal
- The busy, done, and all error flags for both channels
- Interrupt requests for both channels
- Single-bit and double-bit error counts

Bit $2^3$ forces the Packet signal to clear. Set this bit after the final activation in a multiple-activation packet.

Bit $2^8$ prevents the Packet signal from clearing between activations. Set this bit to transmit a multiple-activation packet.

Modifier bit M0 determines which channel is affected by bit $2^8$; it does not affect bits $2^0$ through $2^3$.

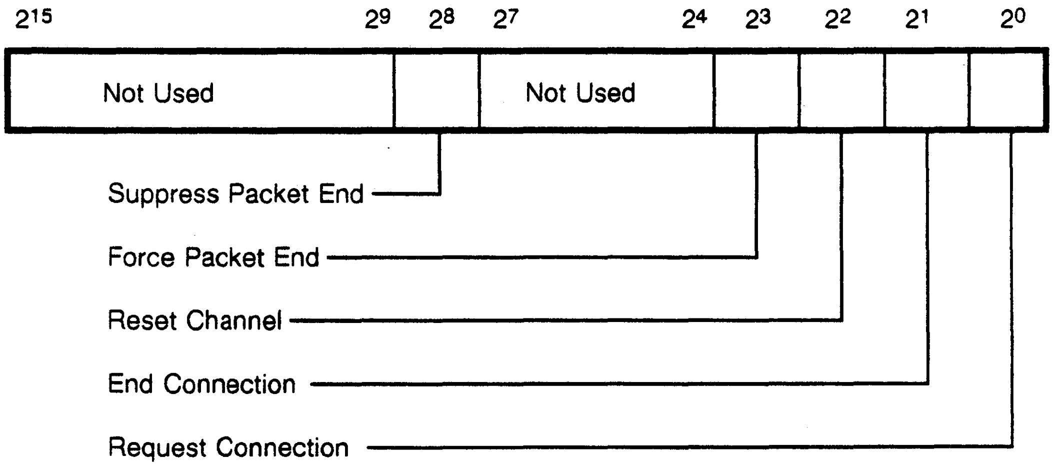


Figure 5-22. HCA4:4 Control Parcel

### HCA4:5 - Write Halfword Count and Activate Channel

This function uses the accumulator contents as the halfword count for the transfer. (In 200-Mbyte/s mode, all transfers contain an even number of halfwords; accumulator bit $2^0$ is ignored.) An accumulator value of 0 causes a transfer of 65,536 halfwords (32,768 words). Modifier bit M0 determines which buffer pointer is used.

If no transfer using the other buffer pointer is in progress, this function starts a transfer immediately. If a transfer using the other pointer is in progress, the transfer is delayed (stacked) until the other transfer completes.

### HCA4:6 - Clear Interrupt Enable Flag

This function clears the channel interrupt enable flag. Modifier bit M0 determines which channel is affected.

### HCA4:7 - Set Interrupt Enable Flag

This function clears the channel interrupt enable flag. Modifier bit M0 determines which channel is affected.

### HCA4:12 - Read Halfword Count or Status

This function performs either of two operations, depending on the value of accumulator bit $2^0$. Modifier bit M0 is not used.

If accumulator bit $2^0$ is 0, this function transfers the contents of the channel status register to the accumulator. Bits $2^0$ through $2^{4,}$ $2^7$ and $2^{15}$ indicate the cause of the most recent interrupt. Bits $2^5$, $2^6$, $2^8$, and $2^9$ provide general status information, and bits $2^{10}$ through $2^{14}$ indicate the state of channel signals.

Table 5-15 lists the function of each bit. If accumulator bit $2^0$ is 1, this function transfers the halfword count to the accumulator.

### HCA4:13 - Read Error Code

This function transfers the contents of the SECDED error code register to the accumulator. Figure 5-23 shows the format of this register. Modifier bit M0 is not used.

Bits $2^0$ through $2^7$ are the syndrome of the first single-bit error. If no single-bit error has occurred, the syndrome corresponds to the most recent double-bit error. If no errors of either type have occurred, the syndrome is 0.

Bits $2^8$ through $2^{11}$ and $2^{12}$ through $2^{15}$ are the number of single-bit and double-bit errors that have occurred.

The syndrome and counters are cleared each time this function is used. They can also be cleared by resetting the channel. (Refer to function HCA4:4.)

### HCA4:17 - Write Diagnostics Modes

This function uses accumulator bits $2^0$ through $2^2$ to select three diagnostic modes. Figure 5-24 shows the function of these accumulator bits.

The LLRC diagnostics mode causes the I field to replace the LLRC transmitted at the end of each burst. In 200-Mbyte/s mode, it also causes functions HCA4:2 and HCA4:3 to write to bits $2^{32}$ through $2^{63}$ of the I field.

The forced parity mode causes I-field bits to replace the parity bits transmitted with each halfword or word. Table 5-16 shows the source of each parity bit in this mode.

The disable 200-Mbyte/s mode causes the incoming Interconnect-B signal to be treated as false, regardless of its actual state. The outgoing Interconnect-B signal is not affected. This function must not be used during a connection.

Table 5-15. HCA4:12 Status Parcel

| Accumulator Bit | Description |
|---|---|
| $2^0$ | Connection accepted interrupt † |
| $2^1$ | Connection rejected interrupt |
| $2^2$ | Connection aborted interrupt |
| $2^3$ | Transfer completed (pointer a) interrupt |
| $2^4$ | Transfer completed (pointer b) interrupt |
| $2^5$ | Activation stacked (pointer a) |
| $2^6$ | Activation stacked (pointer b) |
| $2^7$ | Connection ended interrupt |
| $2^8$ | SECDED error |
| $2^9$ | Buffer available in destination |
| $2^{10}$ | Interconnect-A signal in |
| $2^{11}$ | Interconnect-B signal in |
| $2^{12}$ | Request signal out |
| $2^{13}$ | Connect signal in |
| $2^{14}$ | Packet signal out |
| $2^{15}$ | Interconnect transition interrupt |

† Occurs when either of the following conditions occurs: the Connect signal sets and remains set for 16 HIPPI clock periods; the Connect signal sets, and then the Ready signal sets.

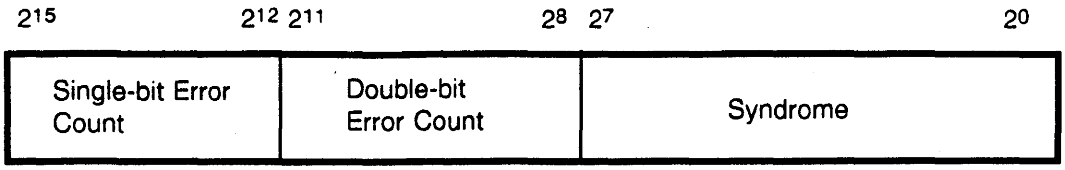


Figure 5-23. HCA4:13 Status Parcel

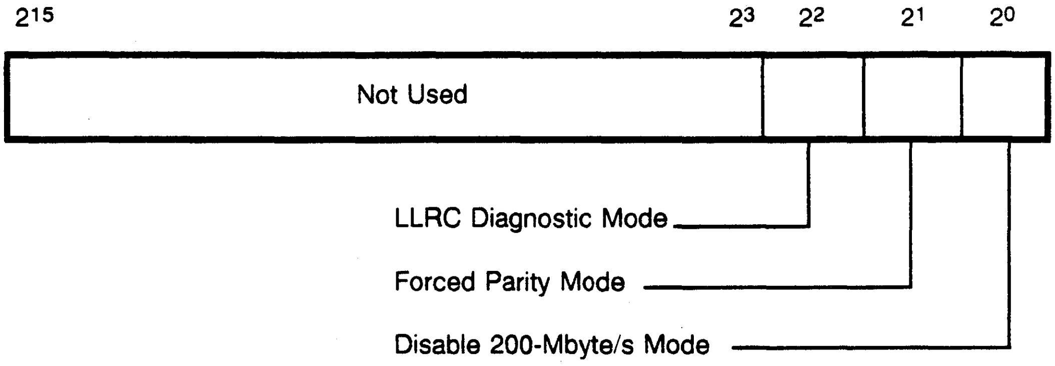


Figure 5-24. HCA4:17 Control Parcel

Table 5-16. Forced Parity Mode Bits

| Parity Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| I Field Bit | $2^{0}$ | $2^{8}$ | $2^{16}$ | $2^{24}$ | $2^{32}$ | $2^{40}$ | $2^{48}$ | $2^{54}$ |

# TCA-1 CHANNEL ADAPTER

The TCA-1 tape channel adapter enables the IOS-E to communicate with IBM-compatible magnetic tape drives. Each TCA-1 can control up to 256 tape drives, but due to timing and electrical considerations, the number of tape drives is generally limited to 8.

## Data Transfers

The TCA-1 transfers data to or from an external device in interlock mode at either 3.0 Mbytes/s or 4.5 Mbytes/s. The TCA-1 operates in a half-duplex mode; that is, data transfers can take place in only one direction at a time. Each data transfer is from 1 to 65,536 bytes long; each byte is 8 bits plus 1 parity bit.

## Channel Signals

Refer to Figure 5-25 for an illustration of the TCA-1 channel signals. The TCA-1 channel signals and channel signal definitions are also provided.

Signal — Definition

Bus 0 Out — The Bus 0 Out signal passes address, data, commands, and status from the channel adapter to the external device.

Bus 0 In — The Bus 0 In signal passes address, data, commands, and status to the channel adapter from the external device.

Address Out — The Address Out signal is raised by the channel adapter when there is an address on the Bus Out line.

Address In — The Address In signal is raised by the external device when it returns the address to the channel adapter. The address is returned by the selected external device.

Command Out — The Command Out signal is raised by the channel adapter when there is a command on the Bus Out line, or the byte count equals zero.

Status In — The Status In signal is raised by the external device when it is inputting status to the channel adapter. The external device sends one byte of initial status and one byte of ending status.

Service Out — The Service Out signal is raised by the channel adapter for each byte of data, or is raised in response to a Status In signal.

Service In — The Service In signal is raised by the external device when it is sending or receiving a byte of data.

Data Out — The Data Out signal is raised by the channel adapter when it is accepting or sending a byte of data.

Signal Definition

Data In The Data In signal is raised by the external device when it is accepting or sending a byte of data.

Operational Out The Operational Out signal enables the control signals for the channel adapter and external device.

Hold Out The Hold Out signal is used with Select Out to select the external device.

Select Out The Select Out signal is used with Hold Out to select the external device.

Operational In The Operational In signal indicates that the external device is selected, and is active.

Suppress Out The Suppress Out signal suppresses data, status, chaining command, and selective reset.

Select In The Select In signal is maintained if no external device selects the address.

Request In The Request In signal indicates that the external device needs service.

Disconnect In The Disconnect In signal is raised by the external device if an error occurred.

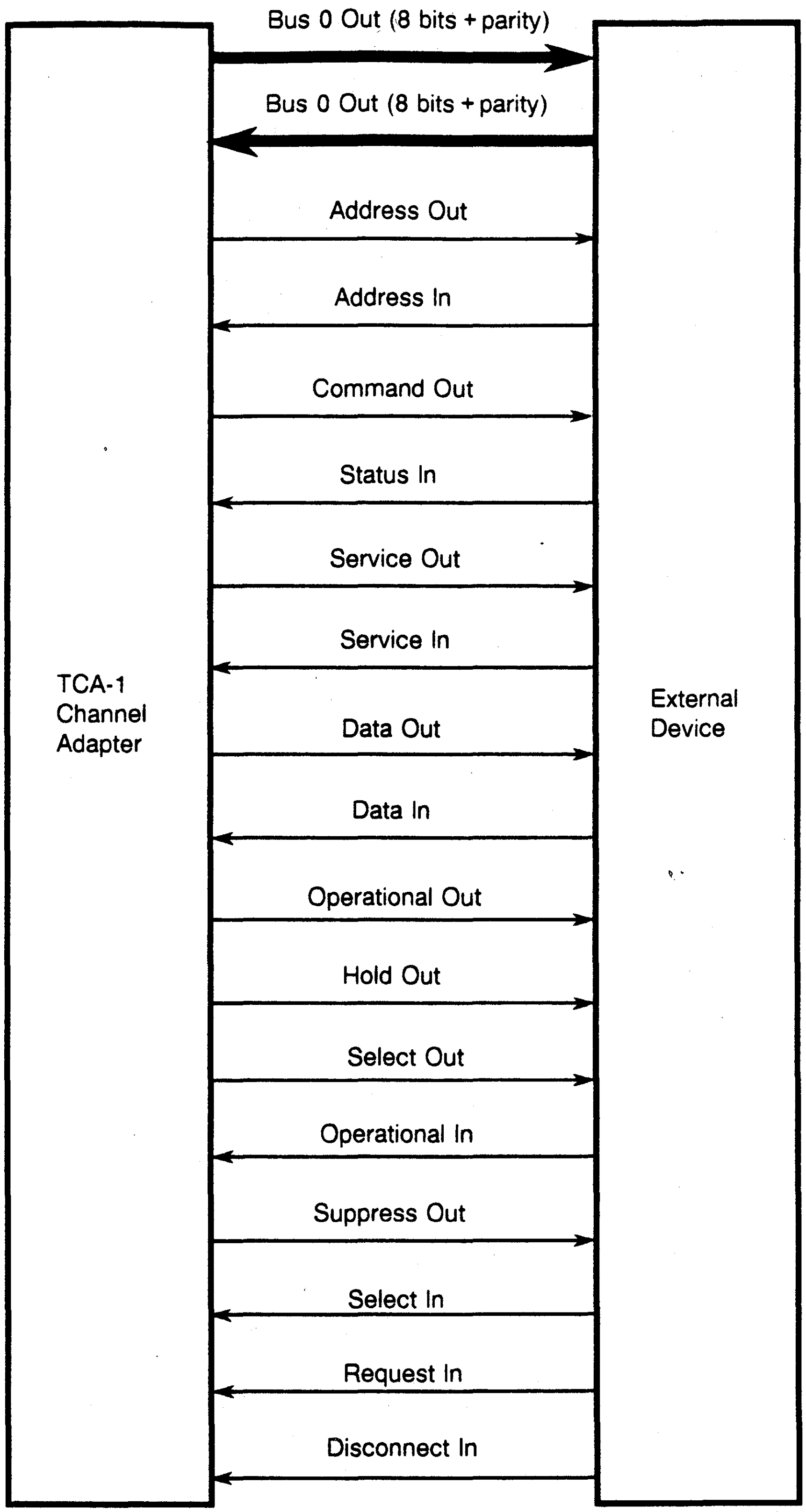


Figure 5-25. TCA-1 Channel Signals

## Channel Functions

The functions can have slightly different uses, depending on whether the function is issued on an even-numbered channel or an odd-numbered channel. Also, the settings of the *d* field and B register mode bits as well as accumulator bits can affect the results of some of the functions. Table 5-17 lists the functions and the ways each function can be used.

Table 5-17. TCA-1 Channel Functions

| Function | Description |
|---|---|
| TCA1:0 | Clear channel |
| TCA1:1 | Start transfer<br>Start input transfer (even channel)<br>M2, M1, M0 = 001: read status data<br>M2, M1, M0 = 011: read sequencer program<br>M2, M1, M0 = 101: read ECC registers<br>Start output transfer (odd channel)<br>M1, M0 = 00: write control data<br>M1, M0 = 10: write sequencer program |
| TCA1:2 | Not used |
| TCA1:3 | Not used |
| TCA1:4 | Write byte count (0 through 65,536) |
| TCA1:5 | Write starting sequencer address and I register |
| TCA1:6 | Clear interrupt enable flag |
| TCA1:7 | Set interrupt enable flag |
| TCA1:10 | Read local memory address |
| TCA1:11 | Read inverted local memory parcel count |
| TCA1:12 | Read status 0 |
| TCA1:13 | Read status 1 |
| TCA1:14 | Write local memory starting address |
| TCA1:15 | Write local memory parcel count |
| TCA1:16 | Write control registers j and k |
| TCA1:17 | Even channel: write mode selects<br>Odd channel: write bit stream number † |

† Not used.

### TCA1:0 - Clear Channel

This function aborts any transfer in progress and clears the channel busy and done flags. If modifier bits M1, M2, and M3 are set, this function also performs a master clear of the channel adapter.

### TCA1:1 Start Transfer

This function initiates a data transfer from the channel adapter to IOP local memory, or a data transfer from IOP local memory to the channel adapter. Refer to the following subsections for information on the TCA1:1 input and output transfer functions.

#### TCA1:1 - Start Input Transfer (Even Channel)

This function initiates a data transfer from the channel adapter to IOP local memory. Functions TCA1:14 and TCA1:15 must be issued before this function to set up the local memory address and parcel count. Function TCA1:0 must be issued after this function to clear the channel adapter's control logic and buffered data.

Modifier bits M0 through M2 affect the operation of this function. Bit M0 must be 1; this causes the channel adapter to use the busy, done, and interrupt flags for the odd channel while the IOP DMA control logic uses the flags for the even channel. Using both sets of flags allows the interface to be monitored separately from the DMA channel.

The operation of this function for the three defined combinations of M0 through M3 is shown below:

| M3 | M2 | M1 | M0 | Operation |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | Read status data. This function is used in conjunction with a device command. It works similarly to function TCA1:5, except that data is transferred to local memory. When this function issues, the accumulator must contain the sequencer starting address and a value for the i register (the i register loads commands sent to a drive). Refer to function TCA1:5 for the accumulator format. |
| 0 | 0 | 1 | 1 | Read sequencer program. This function transfers the contents of the sequencer RAM (1,024 words × 32 bits) to local memory. Bits $2^{16}$ through $2^{31}$ of each word are transferred first, followed by bits $2^{0}$ through $2^{15}$. |
| 0 | 1 | 0 | 1 | Read ECC registers. This function transfers the contents of eight 4-byte ECC registers to local memory. |

#### TCA1:1 - Start Output Transfer (Odd Channel)

This function initiates a data transfer from IOP local memory to the channel adapter. Functions TCA1:14 and TCA1:15 must be issued before this function to set up the local memory address and parcel count. Function TCA1:0 must be issued after this function to clear the channel adapter's control logic and buffered data.

Modifier bits M0 and M1 affect the operation of this function. Bit M0 must be 0; this causes the channel adapter to use the busy, done, and interrupt flags for the even channel while the IOP DMA control logic uses the flags for the odd channel.

The operation of this function for both defined combinations of M0 through M3 is shown below:

| M3 | M2 | M1 | M0 | Operation |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Write control data. The status format will be defined later. This function is used in conjunction with a device command. It works similarly to function TCA1:5, except that data is transferred from local memory. When this function issues, the accumulator must contain the sequencer starting address and a value for the i register; refer to function TCA1:5 for the accumulator format. |
| 0 | 0 | 1 | 0 | Write sequencer program. This function transfers local memory data to the sequencer RAM (1,024 words × 32 bits). Bits $2^{16}$ through $2^{31}$ of each word are transferred first, followed by bits $2^0$ through $2^{15}$. |

### TCA1:4 - Write Byte Count (0 through 65,536)

The accumulator value becomes the byte count for the transfer. The maximum byte count for a tape transfer is 65,535.

### TCA1:5 - Write Starting Sequencer Address and i Register

This function uses the accumulator contents as two fields for the channel adapter. Bits $2^0$ through $2^6$ are the sequence starting address. Bits $2^8$ through $2^{15}$ are transmitted to the i register, which is used for a device address or a bus control code depending on the instruction sequence. Bit $2^7$ is not used. Refer to Figure 5-26 for the accumulator format.

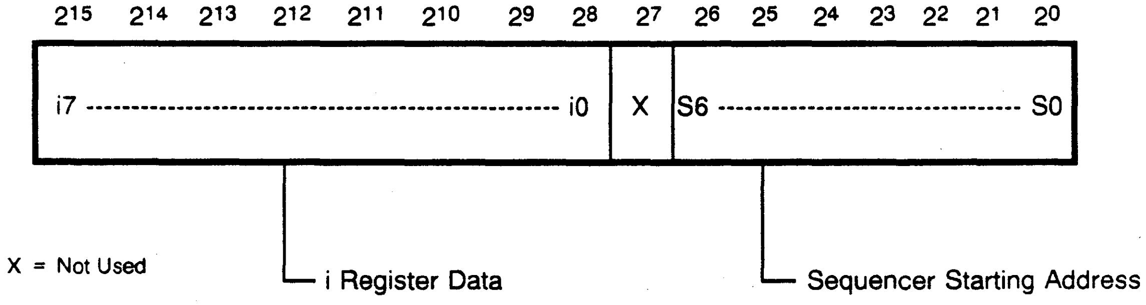

2^15 2^14 2^13 2^12 2^11 2^10 2^9 2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
i7 ---------------------------------------- i0
X
S6 ------------------------- S0
X = Not Used
i Register Data
Sequencer Starting Address

Figure 5-26. TCA:5 Control Parcel

### TCA1:6 - Clear Interrupt Enable Flag

This function clears the channel interrupt enable flag.

### TCA1:7 - Set Interrupt Enable Flag

This function sets the channel interrupt enable flag.

### TCA1:10 - Read Local Memory Address

This function transfers the contents of the local memory address register to the accumulator.

### TCA1:11 - Read Inverted Local Memory Parcel Count

This function transfers the inverted contents of the parcel count register to the accumulator. To obtain the correct parcel count, subtract the accumulator contents from $177777_8$.

### TCA1:12 - Read Status 0

This function transmits one of four status parcels to the accumulator. The contents of accumulator bits $2^0$ and $2^1$ when the function issues determines which status parcel is returned. Refer to Table 5-18 which details all four status parcels.

The even and odd channels return separate status parcels, except for the input tags (Attn-In, Slave-In, and Sync-In). The input tags are read out dynamically for each successive TCA1:12 function.

### TCA1:13 - Read Status 1

This function returns various information to the accumulator, depending on the settings of accumulator bits at the time of function issue. For an accumulator value of 0, this function reads the status byte and the address byte to the accumulator. For an accumulator value of 1, this function reads the input tags from the bus in tag lines. For an accumulator value of 2, this function returns the byte count. Refer to Table 5-19 for the TCA:13 status bits.

### TCA1:14 - Write Local Memory Starting Address

This function transfers the contents of the accumulator to the local memory address register.

### TCA1:15 - Write Local Memory Parcel Count

This function transfers the contents of the accumulator to the parcel count register. The maximum parcel count is equal to $377_8$.

Table 5-18. TCA1:12 Status Parcels

| Bit | Accumulator Bits $2^0$ and $2^1$ | | | |
|---|---|---|---|---|
| | Bits = 00 (binary) | Bits = 01 (binary) | Bits = 10 (binary) | Bits = 11 (binary) |
| $2^0$ | Channel adapter status 0 | Syndrome bit 0 | Transfer count $2^0$ | Service-in |
| $2^1$ | Channel adapter status 1 | Syndrome bit 1 | Transfer count $2^1$ | Data-in |
| $2^2$ | Channel adapter status 2 | Syndrome bit 2 | Transfer count $2^2$ | Status-in |
| $2^3$ | Channel adapter status 3 | Syndrome bit 3 | Transfer count $2^3$ | Request-in |
| $2^4$ | Channel adapter status 4 | Syndrome bit 4 | Transfer count $2^4$ | Operational-in |
| $2^5$ | Channel adapter status 5 | Syndrome bit 5 | Transfer count $2^5$ | Address-in |
| $2^6$ | Channel adapter status 6 | Syndrome bit 6 | Transfer count $2^6$ | Disconnect-in |
| $2^7$ | Channel adapter status 7 | Syndrome bit 7 | Transfer count $2^7$ | Select-in |
| $2^8$ | Single-bit SECDED error | Single-bit error count $2^0$ | Transfer count $2^8$ | Metering-in |
| $2^9$ | Double-bit SECDED error | Single-bit error count $2^1$ | Transfer count $2^9$ | Mark-0-in |
| $2^{10}$ | Voltage fault -4.5 V | Single-bit error count $2^2$ | Transfer count $2^{10}$ | Operational-out |
| $2^{11}$ | Voltage fault +5.0 V | Single-bit error count $2^3$ | Transfer count $2^{11}$ | Address-out |
| $2^{12}$ | Sequencer parity error 0 | Double-bit error count $2^0$ | Transfer count $2^{12}$ | Hold-out |
| $2^{13}$ | Sequencer parity error 1 | Double-bit error count $2^1$ | Transfer count $2^{13}$ | Command-out |
| $2^{14}$ | Sequencer parity error 2 | Double-bit error count $2^2$ | Transfer count $2^{14}$ | Service-out |
| $2^{15}$ | Sequencer parity error 3 | Double-bit error count $2^3$ | Transfer count $2^{15}$ | Suppress-out |

### TCA1:16 - Write Control Registers j and k

This function uses the accumulator contents as two fields for the channel adapter. Bits $2^0$ through $2^7$ are transmitted to the j register. Bits $2^8$ through $2^{15}$ are transmitted to the k register. Refer to Figure 5-27 for the accumulator format.

The sequencer stores bus control commands in the j and k registers.

### TCA1:17 - Write Mode Selects (Even Channel)

This function transmits bits $2^0$ through $2^7$ of the accumulator to the mode select register in the sequencer. Bits $2^8$ through $2^{15}$ are not used. The 8 mode bits can be altered for any function sequence. Refer to Figure 5-28 for the TCA1:17 (even channel) write mode select bits.

Table 5-19. TCA:13 Status Bits

| Accumulator Bit | Accumulator = 0 Ending Status | Accumulator = 1 Input Tags | Accumulator = 2 Byte Count |
|---|---|---|---|
| $2^{0}$ | Unit exception | Not valid | Not valid |
| $2^{1}$ | Unit check | Not valid | Not valid |
| $2^{2}$ | Device end | Not valid | Not valid |
| $2^{3}$ | Channel end | Not valid | Not valid |
| $2^{4}$ | Busy | Not valid | Not valid |
| $2^{5}$ | Control unit end | Not valid | Not valid |
| $2^{6}$ | Status modifier | Not valid | Not valid |
| $2^{7}$ | Attention in | Not valid | Not valid |
| $2^{8}$ | Not used | Not valid | Not valid |
| $2^{9}$ | Not used | Not valid | Not valid |
| $2^{10}$ | Not used | Not valid | Not valid |
| $2^{11}$ | Not used | Not valid | Not valid |
| $2^{12}$ | Not used | Not valid | Not valid |
| $2^{13}$ | Not used | Not valid | Not valid |
| $2^{14}$ | Not used | Not valid | Not valid |
| $2^{15}$ | Not used | Not valid | Not valid |

<table>
<tr><td>$2^{15}$</td><td>$2^{14}$</td><td>$2^{13}$</td><td>$2^{12}$</td><td>$2^{11}$</td><td>$2^{10}$</td><td>$2^{9}$</td><td>$2^{8}$</td><td>$2^{7}$</td><td>$2^{6}$</td><td>$2^{5}$</td><td>$2^{4}$</td><td>$2^{3}$</td><td>$2^{2}$</td><td>$2^{1}$</td><td>$2^{0}$</td></tr>
<tr><td colspan="8">k Register data</td><td colspan="8">j Register data</td></tr>
</table>

Figure 5-27. TCA1:16 Control Register Bits

### TCA1:17 - Enter Bit Stream Number (Odd Channel)

This function is currently not used.

<table>
<tr><th>$2^{15}$</th><th>$2^{14}$</th><th>$2^{13}$</th><th>$2^{12}$</th><th>$2^{11}$</th><th>$2^{10}$</th><th>$2^{9}$</th><th>$2^{8}$</th><th>$2^{7}$</th><th>$2^{6}$</th><th>$2^{5}$</th><th>$2^{4}$</th><th>$2^{3}$</th><th>$2^{2}$</th><th>$2^{1}$</th><th>$2^{0}$</th></tr>
<tr><td colspan="8">Not Used</td><td colspan="8">Mode Select</td></tr>
</table>

Figure 5-28. TCA1:17 (Even Channel) Write Mode Select Bits

# CONTENTS

## FIGURES

## TABLES

# 6. CLUSTER INTERFACE/WORKSTATION INTERFACE

The I/O subsystem (IOS) has two dedicated workstation channel pairs. One channel pair allows the operator workstation (OWS) to communicate with the IOS. The other channel pair allows the maintenance workstation (MWS) to communicate with the IOS. The workstations can use these channels to read IOS status, read or write to local memory in an IOP, and perform maintenance tasks such as deadstarting or master clearing an IOP.

The hardware connections between the workstations and the IOS are shown in Figure 6-1. Each workstation connects to a workstation interface (WIN) through a dedicated 16-bit, 6-Mbyte/s low-speed (LOSP) channel pair. The LOSP channel pair consists of one input and one output channel. Each WIN connects to the eight cluster interfaces (CIN). Each CIN connects to one I/O cluster. This configuration allows two simultaneous data transfers between the workstations and the IOPs.

Each LOSP channel uses 16 data bits to transfer a parcel. Three control signals – Ready, Resume, and Disconnect – control the transfer sequence. The WIN module sends a fourth signal, the Clear Channel signal, if an IOP is requesting I/O, or if a cluster has an error to report. The Clear Channel signal interrupts the workstation if channel interrupts are enabled. Refer to Section 5 of this manual, "Channel Adapters," for more information on the the LOSP data and control signals.

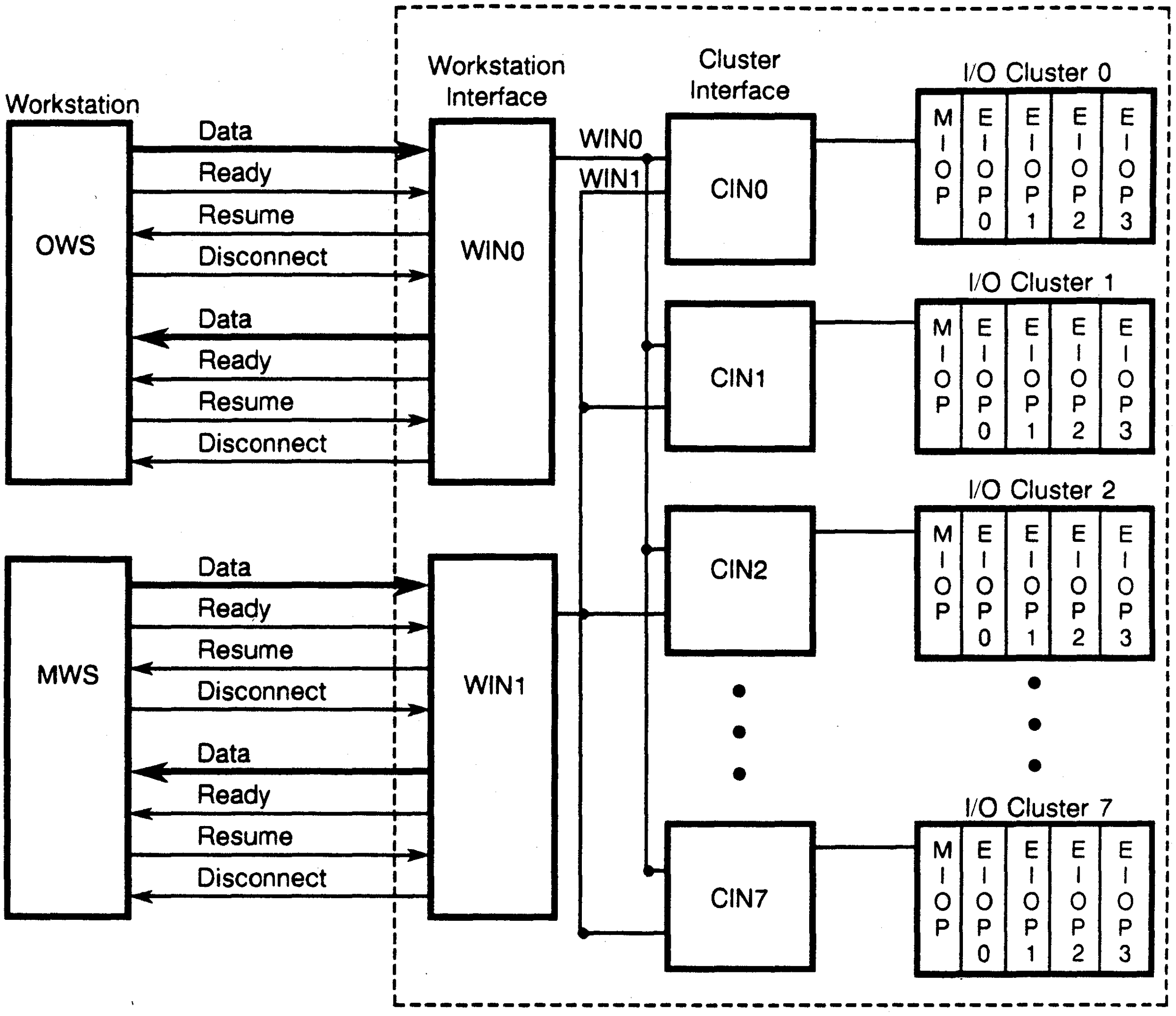
The WINs are programmable but the CIN is not. The workstation is the only device that can send functions to a WIN. When the WIN receives a function, it passes necessary control and data to the CIN. Therefore, operation of the CIN is transparent to the user.

Functions for the CIN/WIN interface are divided into two groups. The first group consists of the workstation-controlled functions. The workstation sends these functions to a WIN to program an IOP, set up and/or initiate a LOSP data transfer, read I/O requests from all IOPs, or perform several maintenance features. The second group consists of the IOP-controlled functions. An IOP uses these functions to program a LOSP channel, to clear channel flags, and to enable/disable interrupt enable flags. An IOP function can also request a LOSP channel transfer. However, the transfer does not begin until the workstation acknowledges the request.

The remainder of this section further explains the workstation-controlled functions, and IOP-controlled functions.

## WORKSTATION-CONTROLLED FUNCTIONS

Workstation-controlled functions control and monitor an IOP and enable a workstation to send data to and receive data from an IOP. This subsection explains the control sequence for workstation-controlled functions and provides detailed descriptions of each function.



The dotted line encloses the components of the IOS.

Figure 6-1. Workstation Connection to an IOP

## Control Sequence

The control sequence for a workstation-controlled function involves a transfer and response header (described in the following subsections). The workstation sends a transfer header to the IOS. The IOS responds to the transfer header by sending a response header back to the workstation. If the workstation-controlled function involves a data transfer, the workstation is now ready to send or receive the data. Refer to the following subsections for additional information on transfer and response headers.

Header data parity errors or control sequence error information is contained in the response header; the workstation must periodically poll the IOS for errors by requesting a return I/O request status from the IOS. (Refer to the "WIN2 - Return I/O Request Status" subsection later in this section.)

## Transfer Headers

The workstation transfer header consists of four parcels. Descriptions of these parcels are provided in the following paragraphs which refer specifically to the WIN0 format. An illustration of the workstation transfer header parcel format is provided in Figure 6-2.

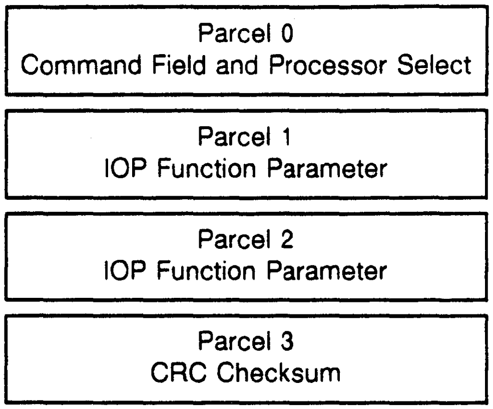


Figure 6-2. Workstation Transfer Header Parcel Format

Parcel 0 contains command, function, cluster select and IOP select bits. This parcel selects EIOP0 through EIOP3 or the multiplexer IOP (IOP MUX) of the specified cluster to decode the function code and perform a function.

Parcels 1 and 2 contain IOP function parameters. If a function involves a data transfer, parcel 1 contains the local memory starting address, and parcel 2 contains the parcel count. If the function does not involve a data transfer, parcels 1 and 2 are not used.

Parcel 3 is a cyclic redundancy check (CRC) checksum that the IOS uses to confirm the validity of the 4 parcels.

The workstation generates the CRC in the following manner:

1. The CRC is initially a workstation memory value of $100000_8$, and a variable Z is initially $004020_8$.

2. If (CRC bit $2^{15}$) XOR (parcel 0 bit $2^{15}$) = 1
   then CRC ← CRC XOR parcel 0 XOR Z,
   else CRC ← CRC XOR parcel 0.

3. Left circular shift CRC one place and replace CRC with the result.

4. If (CRC bit $2^{15}$) XOR (parcel 1 bit $2^{15}$) = 1
   then CRC ← CRC XOR parcel 1 XOR Z,
   else CRC ← CRC XOR parcel 1.

5. Left circular shift CRC one place and replace CRC with the result.

6. If (CRC bit $2^{15}$) XOR (parcel 2 bit $2^{15}$) = 1
   then CRC ← CRC XOR parcel 2 XOR Z,
   else CRC ← CRC XOR parcel 2.

7. Left circular shift CRC one place and replace with the result. This is the CRC that goes into parcel 3.

The WIN regenerates the CRC from parcels 0 through 3 and compares the regenerated CRC with the original CRC. If data is correctly transferred across a channel, the original CRC and the regenerated CRC are identical. If data is incorrectly transferred across a channel, 1 or more bits of the regenerated CRC differ from the original. Using the CRC checksum protects the IOPs from failing workstations and failing I/O channels; the command field of invalid transfers is ignored.

## Response Headers

The response header is sent from the IOS in response to a transfer header sent from the workstation. The response header consists of 4 parcels that contain IOP and I/O cluster status bit information. Refer to Figure 6-3 for an illustration of the response header.

The response header reports status information for up to 8 clusters of IOPs, and performs the following functions.

- Communicates to the workstation whether or not the WIN receives a valid transfer header
- Communicates to the workstation if any of the processors of a cluster are requesting I/O
- Communicates to the workstation if a processor has error data to report
- Communicates to the workstation whether or not a cluster exists
- Communicates to the workstation clock speed information

Some IOSs may have fewer than 8 clusters. If there are fewer than 8 clusters in the system, the missing clusters' status of I/O requests are zero filled.

Bit positions $2^0$ through $2^4$ and $2^8$ through $2^{12}$ represent EIOPs 0 through 3 and the IOP MUX of a cluster. If any of the IOPs of a cluster are requesting output, the corresponding IOP number bit sets in the appropriate parcel.

Bit positions $2^5$ and $2^{13}$ (labeled E) are error-reporting positions. If one or more of the IOPs in a cluster have an error to be reported, the E bit of that cluster is set.

Bit positions $2^6$ and $2^{14}$ (labeled I) indicate whether or not a cluster exists. The I bit is a 0 if the cluster is installed, and is a 1 if a cluster is not installed.

Bit position $2^7$ detects header errors. Bit $2^7$ of parcel 0 sets if the header was invalid because of a data parity error, sequence error or CRC error. Bits $2^7$ is not used in parcels 1 through 3.

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C0 | I | E | 4 | 3 | 2 | 1 | 0 | H | I | E | 4 | 3 | 2 | 1 | 0 | Parcel 0 |
| Clock Select $2^0$ | Cluster 1 Data | | | | | | | Transfer Header Error | Cluster 0 Data | | | | | | | |

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C1 | I | E | 4 | 3 | 2 | 1 | 0 | X | I | E | 4 | 3 | 2 | 1 | 0 | Parcel 1 |
| Clock Select $2^1$ | Cluster 3 Data | | | | | | | | Cluster 2 Data | | | | | | | |

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C2 | I | E | 4 | 3 | 2 | 1 | 0 | X | I | E | 4 | 3 | 2 | 1 | 0 | Parcel 2 |
| Clock Select $2^2$ | Cluster 5 Data | | | | | | | | Cluster 4 Data | | | | | | | |

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 † | I | E | 4 | 3 | 2 | 1 | 0 | X | I | E | 4 | 3 | 2 | 1 | 0 | Parcel 3 |
| | Cluster 7 Data | | | | | | | | Cluster 6 Data | | | | | | | |

X = Not used.

† Reserved for future use.

Figure 6-3. Response Header

Bit position $2^{15}$ of parcels 0 through 2 indicates the selected clock speed. Bit $2^{15}$ of parcel 3 is reserved for future use.

If no data follows the response header, the response header is followed by a Disconnect signal; the IOS activates the Disconnect signal to indicate that the transfer is complete.

### Transfer and Response Header Errors

The following types of transfer and response header errors may occur during a transfer:

- CRC errors
- Parity error on a transfer or response header parcel
- Transfer or response header less than 4 parcels

If any one of these errors occurs, the transfer header function is not executed and is interpreted by the IOS as a return I/O request status. Bits $2^6$ and $2^7$ of parcel 0 set to indicate the error. Additional header functions are interpreted in the same manner until a channel master clear is performed, which initializes the workstation-to-IOS connection.

## Workstation Interface Commands

WIN commands are selected by the user to perform specific tasks. The WIN decodes bits $2^{12}$ through $2^{15}$ of parcel 0 of the transfer header to determine which WIN command is sent by the workstation.

The WIN0 command determines the command, function, cluster, and processor information to be used for a particular IOP function. The selection of WIN commands 2 through 5 determines the commands to be performed by the WIN. Each WIN command is described in the following subsections. Refer to Table 6-1 for a description of the WIN commands.

Table 6-1. WIN Commands

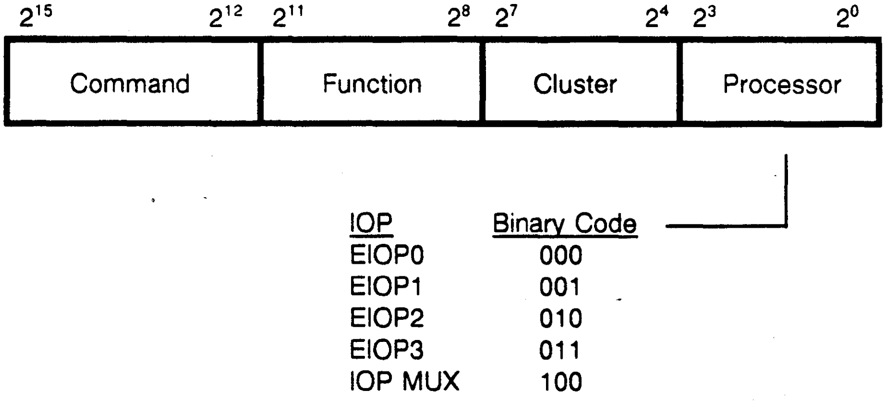
| Command | Parcel 0 Data (binary) | Operation |
|---|---|---|
| WIN0 | 0000 ffff cccc pppp | IOP function |
| WIN2 | 0010 xxxx xxxx xxxx | Return I/O request status |
| WIN3 | 0011 xxxx xxxx xsss | Select clock speed |
| WIN4 | 0100 xxxx xxxx xxxx | Loop-back transfer |
| WIN5 | 0101 xxxx xxxx xxxx | System master clear |

ffff - Function code number
cccc - Cluster number
pppp - Processor number
xxxx - Not used
sss - Clock speed

Some WIN command functions use local memory starting addresses and parcel counts. Each IOP has two independent sets of starting address and parcel count registers. The two sets are fully independent; using one set does not affect the contents of the other set. (Refer to Section 2 of this manual, "I/O Processor," for additional information on IOP registers.)

## WIN0 - IOP Function

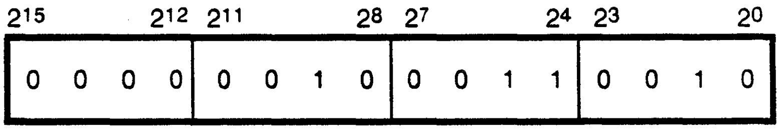
WIN0 is sent from the workstation to the IOP and contains WIN command, function, cluster, and processor information. Refer to Figure 6-4 for an example of a WIN0 command parcel. The WIN0 command passes the transfer header and data to the cluster and to one of five IOPs specified in the command parcel.



| IOP | Binary Code |
|---|---|
| EIOP0 | 000 |
| EIOP1 | 001 |
| EIOP2 | 010 |
| EIOP3 | 011 |
| IOP MUX | 100 |

Figure 6-4. WIN0 Command Parcel

The command parcel can select any of the five IOPs in any cluster. Each IOP is assigned a binary code that, when entered in bits $2^0$ through $2^3$, selects the appropriate IOP. Similarly, binary codes are entered in bits $2^4$ through $2^{15}$ to select an I/O cluster, function, and command. Refer to Figure 6-5 for an example of a WIN0 command parcel. In this example, WIN0, function 2, cluster 3, and EIOP2 are selected.



| $2^{15}$ – $2^{12}$ | $2^{11}$ – $2^{8}$ | $2^{7}$ – $2^{4}$ | $2^{3}$ – $2^{0}$ |
|---|---|---|---|
| 0 0 0 0 | 0 0 1 0 | 0 0 1 1 | 0 0 1 0 |

Figure 6-5. WIN0 Command Parcel Example

WIN0 function codes are listed in Table 6-2, and are further explained in the following subsections.

### WIN0 Function 1 - Enable IOP Input Channel

This function enables an IOP-controlled data transfer from a workstation to an IOP. Parcels 1 and 2 of the header are not used. The local memory starting address and parcel count are generated from the set of registers under IOP control. All parcels after the transfer header are written into local memory.

Table 6-2. WIN0 Function Codes

| Octal Function Codes † | Purpose |
|---|---|
| 1 | Enable IOP input channel |
| 2 | Write local memory |
| 3 | Deadstart IOP local memory |
| 4 | Read error register |
| 5 | Enable IOP output channel |
| 6 | Read local memory |
| 10 | Master clear |
| 11 | Cluster master clear |
| 14 | Disable write check bits |
| 15 | Enable write check bits |
| 16 | Disable instruction error hang |
| 17 | Enable instruction error hang |

† Function codes 0, 7, 12, and 13 are not used.

### WIN0 Function 2 - Write Local Memory

This function transfers data into the IOP local memory under workstation control. The IOP operation is not affected by this transfer; however, local memory access conflicts may be more frequent. Parcel 1 of the transfer header contains the starting address and parcel 2 contains the parcel count. All parcels after the transfer header are written to local memory. A Disconnect signal terminates the transfer but does not generate an interrupt request.

### WIN0 Function 3 - Deadstart IOP Local Memory

This function loads deadstart routines into the IOP local memory through the IOP input channel. Parcel 1 of the transfer header contains the starting address and parcel 2 contains the parcel count. All parcels after the transfer header are written into local memory.

### WIN0 Function 4 - Read Error Register

This function reads the local memory error register. Transfer header parcels 1 and 2 are not used.

The selected IOP responds by sending 2 parcels of data to the workstation input channel. The first parcel is error data as shown in Figure 6-6. The single-bit (bit $2^0$) error flag sets if syndrome bits are generated for a single-bit error. The double-bit (bit $2^1$) error flag sets if syndrome bits are generated for a double-bit error. Both error flags may simultaneously be set; however, the first single-bit syndrome to occur is written into the syndrome register (bits $2^2$ through $2^7$). Bits $2^8$ through $2^{12}$ of parcel 1, when set, indicate local memory address, exit stack, instruction stack, or upper and lower operand parity errors. Bits $2^{13}$ through $2^{15}$ are not used.

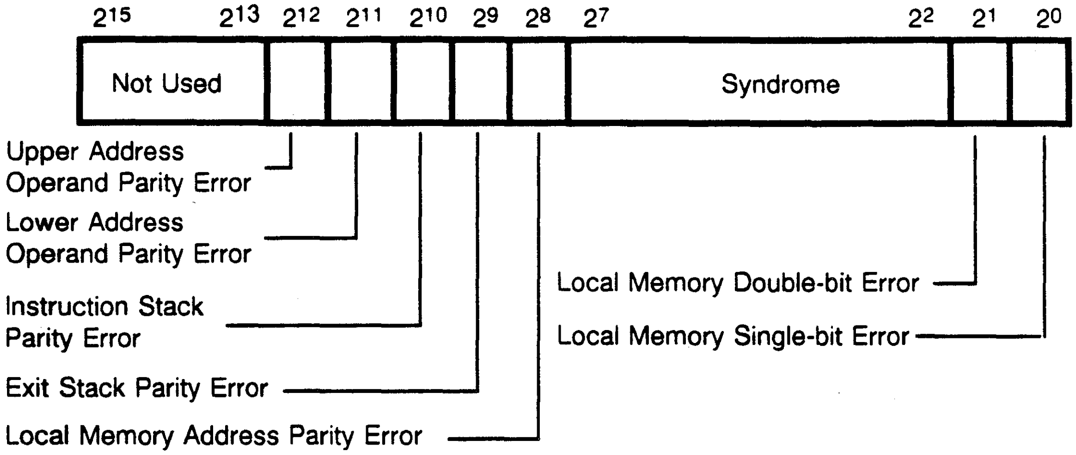

2^15
2^13
2^12
2^11
2^10
2^9
2^8
2^7
2^2
2^1
2^0
Not Used
Syndrome
Upper Address Operand Parity Error
Lower Address Operand Parity Error
Instruction Stack Parity Error
Exit Stack Parity Error
Local Memory Address Parity Error
Local Memory Double-bit Error
Local Memory Single-bit Error

Figure 6-6. WIN0 Function 4 Error Information Parcel

The error register is zeroed out after sending parcel 1. The second response parcel is identical in format to parcel 1 and performs the same function as parcel 1. However, since the error register is zeroed out when the first response parcel is sent, parcel 2 contains all 0's.

If a second error occurs after the first response parcel is sent, the second response parcel contains the error information.

### WIN0 Function 5 - Enable IOP Output Channel

This function enables an IOP-controlled data transfer from an IOP to a workstation and is sent after an IOP requests a transfer. Parcels 1 and 2 of the header are not used. The local memory starting address and parcel count are generated from the set of registers under IOP control. All parcels after the transfer header are written into local memory.

### WIN0 Function 6 - Read Local Memory

This function starts a workstation-controlled data transfer from an IOP to the workstation. Header parcel 1 contains the local memory starting address and parcel 2 contains the parcel count. The WIN sends a Disconnect signal to the workstation after the requested number of parcels are sent. An interrupt request is not generated when the transfer is complete.

### WIN0 Function 10 - Master Clear

This function sends a Master Clear signal to the selected IOP and its associated channel adapters. If the IOP MUX of a cluster is selected, a Master Clear signal is also sent to the high-speed (HISP) and LOSP channels of the IOP MUX.

### WIN0 Function 11 - Cluster Master Clear

This function sends a Master Clear signal to all IOPs, channel adapters, LOSP channels, and HISP channels of the selected cluster.

### WIN0 Function 14 - Disable Write Check Bits

This function disables the writing of check bits into local memory. This function is used for diagnostic purposes to verify a single-error correction/double-error detection (SECDED) operation and error reporting. A WIN0 function 15 command or a Master Clear signal must be issued to re-enable the writing of check bits.

### WIN0 Function 15 - Enable Write Check Bits

This function enables local memory check bit writing. A Master Clear signal also enables check bit writing.

### WIN0 Function 16 - Disable Instruction Error Hang

The exit and instruction stacks are protected with a parity generation and checking circuit. This function disables the instruction error hang function. Diagnostic programs are allowed to continue running even though an instruction parity error is detected in the exit or instruction stack. A WIN0 function 17 command or a Master Clear signal must be issued to re-enable the instruction error hang function.

Refer to the program exit and program instruction stack subsections in Section 2 of this manual, "I/O Processor," for additional information on exit and instruction stack parity errors.

### WIN0 Function 17 - Enable Instruction Error Hang

This function re-enables the instruction error hang function. Diagnostic programs are not allowed to continue running if an instruction parity error is detected in the exit or instruction stack. If an instruction parity error is detected, parity error and interrupt enable flags set.

## WIN2 - Return I/O Request Status

This command returns the response header to the workstation and performs no other function. Parcels 1 and 2 of the transfer header are not used. Refer to the subsection on response headers provided earlier in this section for more information on response headers.

### WIN3 - Select Clock Margin

This command enables the WIN to set a clock speed for the entire IOS for diagnostic purposes. The clock margins can be set from the MWS only; WIN3 commands sent by the OWS are ignored. Parcels 1 and 2 of the transfer header are not used.

Three bits in the command (bits $2^0$ through $2^2$) select the clock speed as shown in Figure 6-7. Bits $2^3$ through $2^{11}$ of the WIN3 clock speed command parcel are not used.

The states of the clock select bits are reported in bit $2^{15}$ of parcels 0, 1 and 2 of the response header. Clock select bit 0 is in bit $2^{15}$ of parcel 0, clock select bit 1 is in bit $2^{15}$ of parcel 1, and clock select bit 2 is in bit $2^{15}$ of parcel 2.

| $2^{15}$ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | x | x | x | x | x | x | x | x | x | c | c | c |

c = Clock Speed
x = Not Used

| Octal value | Clock Speed |
|---|---|
| 0 | 160 MHz (normal) |
| 1 | Undefined |
| 2 | Undefined |
| 3 | Undefined |
| 4 | 152 MHz (slow) |
| 5 | Undefined |
| 6 | External Oscillator |
| 7 | 168 MHz (fast) |

Figure 6-7. WIN3 Clock Speed Command Parcel

### WIN4 - Loop-back Transfer

This command returns workstation data to the workstation for maintenance purposes. The loop-back transfer path is internal to the WIN; data is not sent to the CIN or any other IOPs. The 4-parcel transfer header is sent to the WIN, followed by data parcels. The WIN then returns a 4-parcel I/O request response to the workstation, along with the previously received transfer header and data parcels.

Parcel 3 of the returned transfer header contains the original parcel 2 contents rather than the original CRC parcel; the rest of the received data is returned to the workstation without being changed.

### WIN5 - System Master Clear

This command sends a Master Clear signal to all IOPs, channel adapters, HISP/LOSP channels, and I/O buffers of all I/O clusters. Parcels 1 and 2 of the transfer header are not used.

# IOP-CONTROLLED FUNCTIONS

IOP-controlled functions enable the IOP to send data back and forth between the IOP and the OWS or MWS. These functions can be used to program a LOSP channel, clear channel flags, and enable/disable interrupt enable flags. An IOP function can request a LOSP channel transfer; however, the transfer does not begin until the workstation acknowledges the request.

The IOP-controlled functions are divided into two groups: input and output. The input and output functions are described in the following subsections.

## I/O Processor Input Functions

The IOP input functions control the transfer of data from an OWS or MWS to local memory of an IOP. Each IOP in the IOS has two identical channel pairs that connect to the OWS and MWS. These channel pairs operate independently and can be active simultaneously. Channel 22 is the input channel to the IOP from the OWS; channel 24 is the input channel to the IOP from the MWS. Refer to Table 6-3 for a list of the IOP channel input functions.

Table 6-3. IOP Input Functions

| Function | Description |
|---|---|
| WIA:0 | Clear channel |
| WIA:1 | Start transfer |
| WIA:3 | Clear parity error flags |
| WIA:6 | Clear interrupt enable flag |
| WIA:7 | Set interrupt enable flag |
| WIA:10 | Read local memory address |
| WIA:11 | Read inverted parcel count |
| WIA:13 | Read status |
| WIA:14 | Write local memory address register |
| WIA:15 | Write local memory parcel count |

### WIA:0 - Clear Channel

This function aborts any transfer in progress and clears the busy and done flags.

### WIA:1 - Start Transfer

This function sets the busy flag, clears the done flag, and starts a transfer. This function is used in conjunction with the send data to IOP input channel (WIN0 Function 1) command.

### WIA:3 - Clear Parity Error Flags

This function clears the channel parity error flags.

### WIA:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag.

### WIA:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag.

### WIA:10 - Read Local Memory Address

This function transfers the contents of the local memory address register to the accumulator.

The value in the local memory address register is one greater than the address of the last parcel transferred from local memory.

### WIA:11 - Read Inverted Parcel Count

This function transfers the inverted contents of the parcel count register to the accumulator. Upon completion of the transfer, the done flag sets, and the value of the parcel count register equals 0.

**Note:** To obtain the correct parcel count, subtract the accumulator contents from $177777_8$.

### WIA:13 - Read Status

This function transfers the contents of the status register into the accumulator. Refer to Figure 6-8 for an illustration of the IOP input status bits. Status register bits $2^0$ through $2^2$ and $2^4$ through $2^7$, when set, report the status of a particular IOP or cluster. Bit $2^3$ is not used.

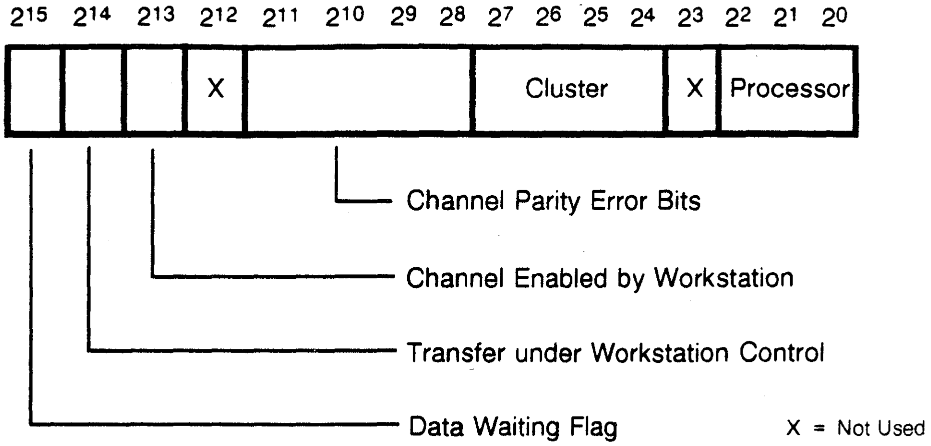


Figure 6-8. WIA:13 IOP Input Status Bits

Bits $2^8$ through $2^{11}$ report channel parity errors. Bit $2^8$, when set, reports channel parity errors for bits $2^0$ through $2^3$. Bit $2^9$, when set, reports channel parity errors for bits $2^4$ through $2^7$. Bit $2^{10}$, when set, reports channel parity errors for bits $2^8$ through $2^{11}$. Bit $2^{11}$, when set, reports channel parity errors for bits $2^{13}$ through $2^{15}$. Bit $2^{12}$ is not used.

Bits $2^{13}$, $2^{14}$, and $2^{15}$ report specific data transfer information. Bit $2^{13}$ sets if the workstation enables the IOP input channel. Bit $2^{14}$ sets if the transfer in progress or the most recently completed transfer used the LMA and parcel count supplied by the workstation. Bit $2^{15}$ sets if the parcel count decrements to 0 and additional data is sent by the workstation. The channel can be reactivated to allow the held data parcel and further data to be transferred. The data waiting flag clears when the channel is reactivated by a data transfer.

### WIA:14 - Write Local Memory Address Register

This function transfers the contents of the accumulator to the input channel local memory address register.

### WIA:15 - Write Local Memory Parcel Count

This function transfers the accumulator contents to the parcel count register; the maximum parcel count is 64 Kwords.

## I/O Processor Output Functions

The IOP output functions control the transfer of data from local memory of an IOP to an OWS or MWS. Each IOP in the IOS has two identical channel pairs that connect to the OWS and MWS. These channel pairs operate independently and can be active simultaneously. Channel 23 is the output channel to the OWS from the IOP; channel 25 is the output channel to the MWS from the IOP. Refer to Table 6-4 for a list of the OWS/MWS channel output functions.

Table 6-4. IOP Output Functions

| Function | Description |
|---|---|
| WOA:0 | Clear channel |
| WOA:1 | Start transfer |
| WOA:6 | Clear interrupt enable flag |
| WOA:7 | Set interrupt enable flag |
| WOA:10 | Read local memory address |
| WOA:11 | Read inverted parcel count |
| WOA:13 | Read status |
| WOA:14 | Write local memory address register |
| WOA:15 | Write local memory parcel count |
| WOA:17 | Write signal control register |

### WOA:0 - Clear Channel

This function aborts any transfer in progress and clears the busy and done flags.

### WOA:1 - Start Transfer

This function sets the busy flag, clears the done flag, and starts a transfer. This function is used in conjunction with the send data to IOP input channel (WINO Function 1) command.

### WOA:6 - Clear Interrupt Enable Flag

This function clears the interrupt enable flag.

### WOA:7 - Set Interrupt Enable Flag

This function sets the interrupt enable flag.

### WOA:10 - Read Local Memory Address

This function transfers the contents of the local memory address register to the accumulator.

The value in the local memory address register is one greater than the address of the last parcel transferred from local memory.

### WOA:11 - Read Inverted Parcel Count

This function transfers the inverted contents of the IOP-controlled parcel count register to the accumulator. Upon completion of the transfer, the done flag sets, and the value of the parcel count register equals 0.

**Note:** To obtain the correct parcel count, subtract the accumulator contents from $177777_8$.

### WOA:13 - Read Status

This function transfers the contents of the status register to the accumulator. Refer to Figure 6-9 for an illustration of the IOP output status bits. Status register bits $2^0$ through $2^2$ and $2^4$ through $2^7$, when set, report the status of a particular IOP or cluster. Bits $2^3$, $2^8$ through $2^{12}$, and $2^{15}$ are not used.

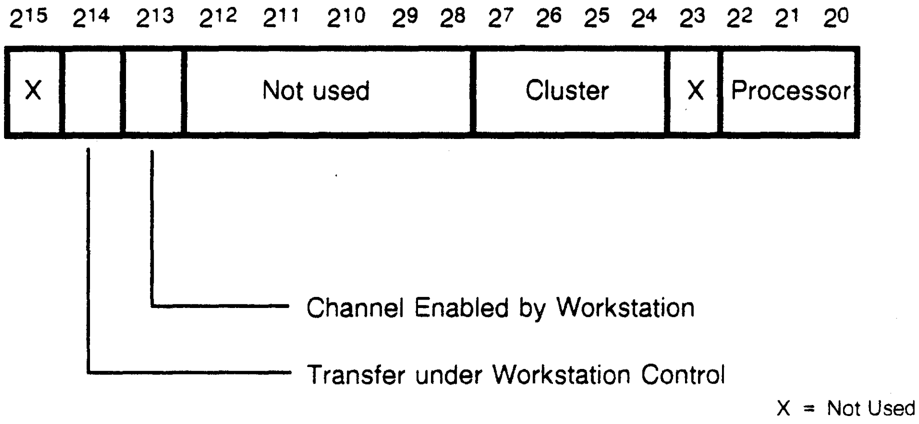


Figure 6-9. WOA:13 IOP Output Status Bits

Bits $2^{13}$ and $2^{14}$ report specific data transfer information. Bit $2^{13}$ sets if the workstation has enabled the IOP output channel. Bit $2^{14}$ sets if the transfer in progress or the most recently completed transfer used the LMA and parcel count supplied by the workstation.

### WOA:14 - Write Local Memory Address Register

This function transfers the contents of the accumulator and loads the parcel into the local memory address register.

### WOA:15 - Write Local Memory Parcel Count

This function transfers the accumulator contents to the parcel count register; the maximum parcel count is 64 Kwords.

### WOA:17 - Write Signal Control Register

This function transfers the contents of the accumulator to the signal control register. The signal control register contains 2 bits that control the Disconnect signal (bits $2^0$ and $2^1$).

If bit $2^0$ is set, no Disconnect signal is sent to the workstation when the parcel count reaches zero at the end of the transfer. If bit $2^0$ is cleared, a Disconnect signal is sent to the workstation when the parcel count reaches zero. If bit $2^1$ is set, the IOS sends a Disconnect signal to the workstation without any transfer of data.

# CONTENTS

**FIGURES**

**TABLES**

# 7. I/O PROCESSOR INSTRUCTION SET

The I/O processor (IOP) instruction set enables a programmer to write programs that perform the following functions:

- Data manipulation - Two's complement add and subtract, logical product calculations, and left, right, circular, and end-off shifts.
- Data transfers - Register to register, local memory to a register, or from a register to local memory.
- Branching - Conditional and unconditional, subroutine calls, and exits.
- Miscellaneous - Disable and enable interrupts, test busy and done flags, and perform I/O functions.

The remainder of this section discusses instruction formats, instruction issue conflicts, accumulator conflicts, register conflicts, and path reservations. Instruction set symbols and instructions 000 through 177 are provided.

## INSTRUCTION FORMATS

All instructions are either 1 or 2 parcels in length. A 1-parcel instruction consists of a 7-bit function code (*f*) field and a 9-bit designator (*d*) field. A 2-parcel instruction contains the *f* and *d* fields and a 16-bit *k* field. Figure 7-1 shows both instruction formats.

The *f*, *d*, and *k* fields perform different functions. The *f* field specifies the instruction to be performed. The *d* field contains a constant that performs several different functions, depending on the instruction designated in the *f* field. The *d* field can be a shift count, a logical constant, a numeric constant, a pointer to an operand register, a branch offset, or an I/O channel designator. The *k* field can be a logical or numeric constant or a jump offset.

## INSTRUCTION ISSUE CONFLICTS

Accumulator and other register conflicts may delay instruction issue. If, for example, a register is in use when an instruction is issued, a conflict occurs and the instruction is held until that register clears. If there are no conflicts with a register, the instruction issues.

The following subsections describe register conflicts in more detail.

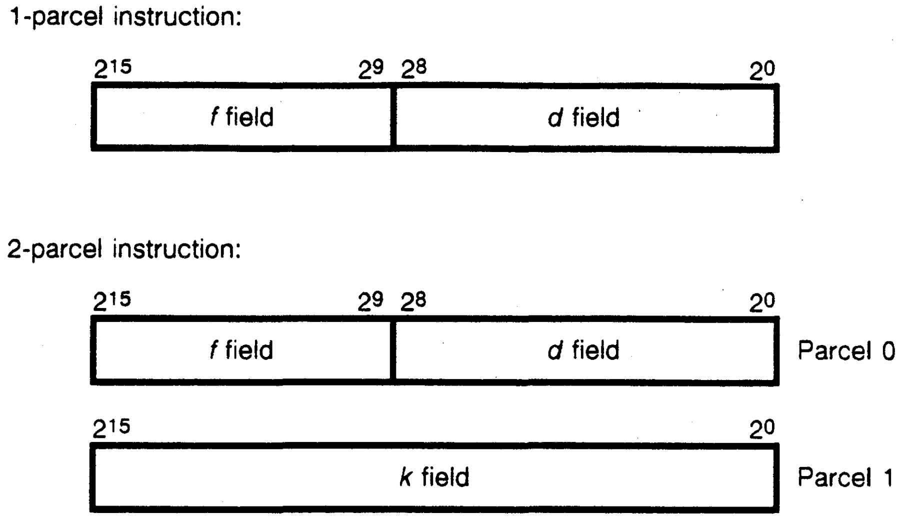


Figure 7-1. Instruction Formats

## Next Instruction Parcel Register Conflicts

Some registers, such as the next instruction parcel (NIP) register, may not be valid at instruction issue. This occurs if the next parcel to be sent to the NIP register is not available from the instruction stack.

## Accumulator Conflicts

Accumulator conflicts can occur when an instruction that accesses the accumulator directly follows an instruction that reserves the accumulator. These conflicts are resolved by holding issue of the accessing instruction until the accumulator is free when such access is required.

The following four types of instructions cause accumulator conflicts.

- Instructions that require the accumulator to be free at issue
- Instructions that require the accumulator to be free 1 clock period (CP) after issue
- Instructions that require the accumulator to be free 2 CPs after issue
- Instructions that require the accumulator to be free 3 CPs after issue

The following instructions require the accumulator to be free at the specified CPs:

| Instructions | Requirement |
|---|---|
| 004 through 017, 024, 044 through 045, 047 through 057 | Free at issue |
| 020 through 023, 025 through 027, 034, 046, 060 through 067 | Free 1 CP after issue |
| 040 through 043 | Free 2 CPs after issue |
| 030 through 033, 035 through 037 | Free 3 CPs after issue |

Instructions that use the accumulator as a destination cause the accumulator to be reserved when they issue. The only exceptions are instructions that execute 1 CP after issue; instructions that complete 1 CP after issue do not cause the accumulator to be reserved. For all other instructions, the accumulator becomes free at the indicated CP noted in each instruction description.

## Destination Pointer Conflicts

The destination pointer (DP) register contains a 7-bit pointer that selects one of the operand registers to receive the contents of the accumulator. The DP register also indicates when a write function to a register must occur. If the DP register is full, a subsequent operand register function cannot issue until the write is completed. A register conflict occurs if the DP register is full and an instruction accessing an operand register is ready to issue. The instruction cannot issue until the write operation in progress is completed. If an instruction specifies a no delayed write of an operand register, this means there is no DP register conflict.

## B Register Conflicts

Instructions that reserve the B register for storing results cause B register conflicts. Other instructions that use the B register cannot issue until the results are stored and the B register reservation is released.

## Path Reservation Conflicts

Path reservation conflicts result in conflicts for branch functions. Instruction issue is held if there is a path reservation conflict to or from the adder/shifter.

Some memory or operand functions that use the adder can cause a conflict with the path to the adder. When an I/O function issues, the path from the adder is reserved. Branches do not check the reservation of the accumulator unless they are conditional. This allows branches to issue out of logical step with their equivalent non-branch functions.

# INSTRUCTION ABBREVIATIONS

Refer to Table 7-1 for a list of symbols used in the ELAN (IOS model E language) instruction set. Refer also to Table 7-2 for a list of abbreviations used throughout this section.

Table 7-1. ELAN Instruction Set Symbols

| Symbol | Definition |
|---|---|
| A | Accumulator |
| B | B register |
| [B] | Contents of the operand register addressed by B |
| BZ | Busy |
| C | Carry flag (accumulator bit $2^{16}$) |
| *d* | The *d* field of an instruction (unsigned numeric value < $512_{10}$) |
| *dd* | Operand register addressed by *d* field (*dd* < $128_{10}$) |
| [*dd*] | Memory parcel addressed by contents of operand register *dd* |
| *dd* + *k* | Memory parcel addressed by the sum of operand register *dd* contents and contents of *k* |
| DN | Done |
| E | Exit stack pointer (4 bits) |
| [E] | Exit stack storage location pointed to by E |
| *f* | The *f* field of an IOP instruction |
| I | System interrupt enable flag |
| *k* | The *k* field of an instruction (unsigned numeric value < $65536_{10}$) |
| [*k*] | Memory parcel addressed by the value of *k* |
| P | Program address register |
| R | Return jump program address |
| >! | Shift right end off; carry bit shifts with accumulator |
| <! | Shift left end off; carry bit shifts with accumulator |
| > > | Shift right circular through carry bit |
| < < | Shift left circular through carry bit |
| & | Logical product (AND) |
| = = | Equal |
| != | Not equal |
| iod | 3-character channel mnemonic: IOR, PXS, etc. |

Table 7-2. Miscellaneous Abbreviations and Definitions

| Abbreviation | Definition |
|---|---|
| CIP | Current instruction parcel register |
| CP | Clock period |
| DP | Destination pointer register |
| MA | Memory address register |
| *M*, *N* | Variable instruction cycle times |
| NIP | Next instruction parcel register |
| RP | Reference pointer register |
| -- | Indicates same conditions as the previous CP |

## I/O PROCESSOR INSTRUCTIONS

The remainder of this section provides IOP instructions 000 through 177. All instructions are octal, and are contained in the *f* field of an instruction. Descriptions of each instruction, the results of each instruction, the steps and CPs involved in executing each instruction, and the issue conditions for each instruction are provided. Refer to Table 7-3 for a description of each IOP instruction.

Table 7-3. IOP Instructions

| Machine Instruction | ELAN | Description |
|---|---|---|
| 000 | Pass | No operation |
| 001 | Exit | Exit from subroutine |
| 002 | I=0 | Disable system interrupts |
| 003 | I=1 | Enable system interrupts |
| 004 | A=A>!*d* | Right shift C and A by *d* places, end off |
| 005 | A=A<!*d* | Left shift C and A by *d* places, end off |
| 006 | A=A> >*d* | Right shift C and A by *d* places, circular |
| 007 | A=A< <*d* | Left shift C and A by *d* places, circular |
| 010 | A=*d* | Transmit *d* to A |
| 011 | A=A & *d* | Logical product of A and *d* to A |
| 012 | A=A+*d* | Add *d* to A |
| 013 | A=A - *d* | Subtract *d* from A |
| 014 | A=*k* | Transmit *k* to A |
| 015 | A=A & *k* | Logical product of A and *k* to A |
| 016 | A=A+*k* | Add *k* to A |
| 017 | A=A-*k* | Subtract *k* from A |
| 020 | A=*dd* | Transmit operand register *dd* to A |
| 021 | A=A & *dd* | Logical product of A and operand register *dd* to A |
| 022 | A=A+*dd* | Add operand register *dd* to A |
| 023 | A=A - *dd* | Subtract operand register *dd* from A |
| 024 | *dd*=A | Transmit A to operand register *dd* |
| 025 | *dd*=A+*dd* | Add operand register *dd* to A, result to operand register *dd* |
| 026 | *dd*=*dd*+1 | Transmit operand register *dd* to A, add 1, result to operand register *dd* |
| 027 | *dd*=*dd* - 1 | Transmit operand register *dd* to A, subtract 1, result to operand register *dd* |
| 030 | A=[*dd*] | Transmit contents of memory addressed by operand register *dd* to A |
| 031 | A=A & [*dd*] | Logical product of A and contents of memory addressed by operand register *dd*, result to A |
| 032 | A=A+[*dd*] | Add contents of memory addressed by operand register *dd* to A |

Table 7-3. IOP Instructions (continued)

| Machine Instruction | ELAN | Description |
|---|---|---|
| 033 | A = A - [*dd*] | Subtract contents of memory addressed by operand register *dd* from A |
| 034 | [*dd*] = A | Transmit A to memory addressed by operand register *dd* |
| 035 | [*dd*] = A + [*dd*] | Add memory addressed by operand register *dd* to A, result to same memory location |
| 036 | [*dd*] = [*dd*] + 1 | Transmit memory addressed by operand register *dd* to A, add 1, result to same memory location |
| 037 | [*dd*] = [*dd*] - 1 | Transmit memory addressed by operand register *dd* to A, subtract 1, result to same memory location |
| 040 | C = 1, io*d* = = DN | Set carry equal to 1 if channel *d* done |
| 041 | C = 1, io*d* = = BZ | Set carry equal to 1 if channel *d* busy |
| 042 | C = 1, IOB = = DN | Set carry equal to 1 if channel B done |
| 043 | C = 1, IOB = = BZ | Set carry equal to 1 if channel B busy |
| 044 | A = A > !B | Right shift C and A by B places, end off |
| 045 | A = A < !B | Left shift C and A by B places, end off |
| 046 | A = A > > B | Right shift C and A by B places, circular |
| 047 | A = A < < B | Left shift C and A by B places, circular |
| 050 | A = B | Transmit B to A |
| 051 | A = A & B | Logical product of A and B to A |
| 052 | A = A + B | Add B to A |
| 053 | A = A - B | Subtract B from A |
| 054 | B = A | Transmit A to B |
| 055 | B = A + B | Add B to A, result to B |
| 056 | B = B + 1 | Transmit B to A, add 1, result to B |
| 057 | B = B - 1 | Transmit B to A, subtract 1, result to B |
| 060 | A = [B] | Transmit operand register B to A |
| 061 | A = A & [B] | Logical product of A and operand register B to A |
| 062 | A = A + [B] | Add operand register B to A |
| 063 | A = A - [B] | Subtract operand register B from A |
| 064 | [B] = A | Transmit A to operand register B |

Table 7-3. IOP Instructions (continued)

| Machine Instruction | ELAN | Description |
|---|---|---|
| 065 | [B] = A + [B] | Add operand register B to A, result to operand register B |
| 066 | [B] = [B] + 1 | Transmit operand register B to A, add 1, result to operand register B |
| 067 | [B] = [B] - 1 | Transmit operand register B to A, subtract 1, result to operand register B |
| 070 | P=p+*d* | Jump to p+*d* |
| 071 | P=p - *d* | Jump to p - *d* |
| 072 | R=p+*d* | Return jump to p+*d* |
| 073 | R=p - *d* | Return jump to p - *d* |
| 074 | P=*dd* | Jump to address in operand register *dd* |
| 075 | P=*dd*+*k* | Jump to sum of *k* and operand register *dd* |
| 076 | R=*dd* | Return jump to address in operand register *dd* |
| 077 | R=*dd*+*k* | Return jump to address sum of *k* and operand register *dd* |
| 100 | P=p+*d*, C= =0 | Jump to p+*d* if carry=0 |
| 101 | P=p+*d*, C !=0 | Jump to p+*d* if carry≠0 |
| 102 | P=p+*d*, A= =0 | Jump to p+*d* if A=0 |
| 103 | P=p+*d*, A !=0 | Jump to p+*d* if A≠0 |
| 104 | P=p - *d*, C= =0 | Jump to p - *d* if carry=0 |
| 105 | P=p - *d*, C!=0 | Jump to p - *d* if carry≠0 |
| 106 | P=p - *d*, A= =0 | Jump to p - *d* if A=0 |
| 107 | P=p - *d*, A!=0 | Jump to p - *d* if A≠0 |
| 110 | R=p+*d*, C= =0 | Return jump to p+*d* if carry=0 |
| 111 | R=p+*d*, C!=0 | Return jump to p+*d* if carry≠0 |
| 112 | R=p+*d*, A= =0 | Return jump to p+*d* if A=0 |
| 113 | R=p+*d*, A!=0 | Return jump to p+*d* if A≠0 |
| 114 | R=p - *d*, C= =0 | Return jump to p - *d* if carry=0 |
| 115 | R=p - *d*, C!=0 | Return jump to p - *d* if carry≠0 |
| 116 | R=p - *d*, A= =0 | Return jump to p - *d* if A=0 |
| 117 | R=p - *d*, A!=0 | Return jump to p - *d* if A≠0 |
| 120 | P=*dd*, C= =0 | Jump to address in operand register *dd* if carry=0 |

Table 7-3. IOP Instructions (continued)

| Machine Instruction | ELAN | Description |
|---|---|---|
| 121 | P=*dd*, C!=0 | Jump to address in operand register *dd* if carry ≠0 |
| 122 | P=*dd*, A= =0 | Jump to address in operand register *dd* if A=0 |
| 123 | P=*dd*, A!=0 | Jump to address in operand register *dd* if A≠0 |
| 124 | P=*dd*+*k*, C= =0 | Jump to address in operand register *dd*+*k* if carry =0 |
| 125 | P=*dd*+*k*, C!=0 | Jump to address in operand register *dd*+*k* if carry ≠0 |
| 126 | P=*dd*+*k*, A= =0 | Jump to address in operand register *dd*+*k* if A=0 |
| 127 | P=*dd*+*k*, A!=0 | Jump to address in operand register *dd*+*k* if A≠0 |
| 130 | R=*dd*, C= =0 | Return jump to address in operand register *dd* if carry =0 |
| 131 | R=*dd*, C!=0 | Return jump to address in operand register *dd* if carry ≠0 |
| 132 | R=*dd*, A= =0 | Return jump to address in operand register *dd* if A=0 |
| 133 | R=*dd*, A!=0 | Return jump to address in operand register *dd* if A≠0 |
| 134 | R=*dd*+*k*, C= =0 | Return jump to address in operand register *dd* + *k* if carry=0 |
| 135 | R=*dd*+*k*, C!=0 | Return jump to address in operand register *dd* + *k* if carry≠0 |
| 136 | R=*dd*+*k*, A= =0 | Return jump to address in operand register *dd* + *k* if A=0 |
| 137 | R=*dd*+*k*, A!=0 | Return jump to address in operand register *dd* + *k* if A≠0 |
| 140 to 157 | iod: 0 to 17 | Channel *d* function 0 to 17 |
| 160 to 177 | IOB: 0 to 17 | Channel B function 0 to 17 |

Instruction 000

| Machine Instruction | ELAN | Description |
|---|---|---|
| 000 | Pass | No operation |

**Issue Conditions** None

**Instruction Timing** CP 0 Issue

**Description** Instruction 000 performs no operation.

Instruction 001

| Machine Instruction | ELAN | Description |
|---|---|---|
| 001 | Exit | Exit from subroutine |

**Issue Conditions** None

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue |
| CP 1 | Latch exit function in exit stack |
| CP 2 | Latch contents of location (E)<br>Transmit location (E) bits $2^0$ through $2^7$ to P register |
| CP 3 | Transmit location (E) bits $2^8$ through $2^{15}$ to P register<br>Decrement E<br>Latch parity check |

**Description**

Instruction 001 terminates the current subprogram sequence and returns to the sequence that called the current subprogram. The program exit stack location designated by the E pointer provides the beginning address for the reinitiated sequence. The E pointer then decrements by 1. If the E pointer is at 0 when this instruction issues, then the exit stack underflow flag sets, an interrupt request is generated, and the E pointer remains at 0.

**Note:** Special timing considerations must be accounted for if an Exit instruction follows a program exit stack (PXS) function because it takes 2 to 3 CPs to eliminate conflicts with the E pointer. Allow at least 2 CPs between the issue of an exit function and any of the following PXS functions:

- PXS:1 (Set E pointer to 1)
- PXS:14 (Increment E pointer)
- PXS:15 (Decrement E pointer)

Allow at least 4 CPs between the issue of an exit function and the following PXS function: PXS:16 (Write location E)

Allow at least 6 CPs between the issue of an exit function and either of the following PXS functions:

- PXS:10 (Read status)
- PXS:11 (Read location E)

An add or shift of the accumulator inserted between the modifying function and the exit instruction satisfies all of the clock period conditions specified for the instruction.

Instruction 002

| Machine Instruction | ELAN | Description |
|---|---|---|
| 002 | I = 0 | Disable system interrupts |

**Issue Conditions** None

**Instruction Timing** CP 0 Issue
Clear system interrupt enable flag

**Description**

Instruction 002 clears the system interrupt enable flag.

**Note:** If a 002 instruction follows a 003 instruction, there must be at least a 1-CP delay between the instructions. A PASS (000) instruction between the 002 and 003 instructions meets this requirement.

**Note:** The system interrupt enable flag does not affect program exit stack (PXS) interrupts. PXS channel interrupts can occur if the PXS channel interrupt enable flag is set, regardless of the state of the system interrupt enable flag.

Instruction 003

| Machine Instruction | ELAN | Description |
|---|---|---|
| 003 | I = 1 | Enable system interrupts |

**Issue Conditions** None

**Instruction Timing**

CP 0 Issue

CP 1 Wait for next instruction to be valid in CIP register
Set system interrupt enable flag

**Description**

Instruction 003 sets the system interrupt enable flag. The system interrupt enable flag is set after the next instruction is valid in the current instruction parcel (CIP). The delay in setting the flag allows an interrupt handler program to enable interrupts and then issue an instruction; interrupts are enabled after the exit instruction.

**Note:** If a 002 instruction follows a 003 instruction, there must be at least a 1-CP delay between the instructions. A PASS (000) instruction between the 003 and 002 instructions meets this requirement.

**Note:** The system interrupt enable flag does not affect program exit stack (PXS) interrupts. PXS channel interrupts can occur if the PXS channel interrupt enable flag is set, regardless of the state of the system interrupt enable flag.

**Note:** A 6-CP delay is required after disabling any channel's interrupt enable flag, and after issuing a 003 instruction. This ensures that the channel interrupt request is no longer set when system interrupts are enabled.

Instruction 004

| Machine Instruction | ELAN | Description |
|---|---|---|
| 004 | A = A > !*d* | Right shift C and A by *d* places, end off |

**Issue Conditions** Accumulator free at issue

**Instruction Timing**

CP 0 Issue
Transmit *d* field to addend register

CP 1 Latch *d* field in addend register
Transmit accumulator and addend register contents to shifter

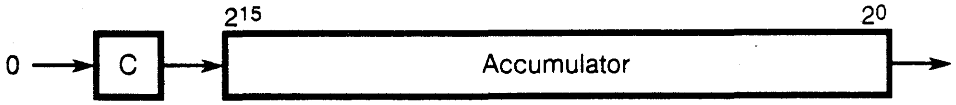
CP 2 First CP in shifter
Invert *d* field
Release addend register

CP 3 Second CP in shifter
Transmit result to accumulator
Release accumulator

CP 4 Latch result in accumulator

**Description**

Instruction 004 shifts the contents of both the accumulator and the carry flag to the right. Bits $2^0$ through $2^4$ of the *d* field provide the shift count; bits $2^5$ through $2^8$ are not used. The carry flag is treated as bit $2^{16}$ of the accumulator. The shift instruction enters 0's into the carry flag and propagates them to the right. No shift is performed if the shift count is 0. A shift count greater than 16 clears the accumulator and carry flag.



Instruction 005

| Machine Instruction | ELAN | Description |
|---|---|---|
| 005 | A=A<!*d* | Left shift C and A by *d* places, end off |

**Issue Conditions** Accumulator free at issue

**Instruction Timing**

CP 0 Issue
Transmit *d* field to addend register

CP 1 Latch *d* field in addend register
Transmit accumulator and addend register contents to shifter

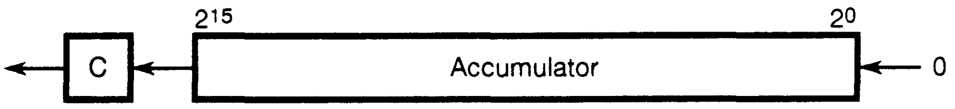
CP 2 First CP in shifter
Release addend register

CP 3 Second CP in shifter
Transmit result to accumulator
Release accumulator

CP 4 Latch result in accumulator

**Description**

Instruction 005 shifts the contents of both the accumulator and the carry flag to the left. Bits $2^0$ through $2^4$ of the *d* field provide the shift count; bits $2^5$ through $2^8$ are not used. The carry flag is treated as bit $2^{16}$ of the accumulator. The shift instruction enters 0's in accumulator bit $2^0$ and propagates them to the left. No shift is performed if the shift count is 0.



Instruction 006

| Machine Instruction | ELAN | Description |
|---|---|---|
| 006 | A=A>>*d* | Right shift C and A by *d* places, circular |

**Issue Conditions** Accumulator free at issue

**Instruction Timing**

CP 0 Issue
Transmit *d* field to addend register

CP 1 Latch *d* field in addend register
Transmit accumulator and addend register contents to shifter

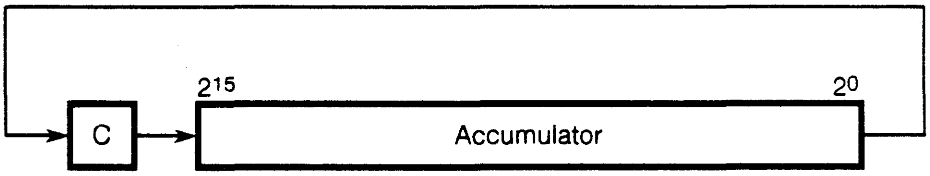
CP 2 First CP in shifter
Invert *d* field
Release addend register

CP 3 Second CP in shifter
Transmit result to accumulator
Release accumulator

CP 4 Latch result in accumulator

**Description**

Instruction 006 shifts the contents of the accumulator and the carry flag to the right in a circular mode. Bits $2^0$ through $2^4$ of the *d* field provide the shift count; bits $2^5$ through $2^8$ are not used. The carry flag is treated as bit $2^{16}$ of the accumulator. No shift is performed if the shift count is 0.



Instruction 007

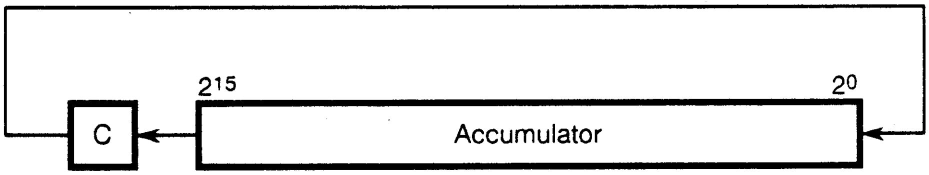
| Machine Instruction | ELAN | Description |
|---|---|---|
| 007 | A=A<<*d* | Left shift C and A by *d* places, circular |

**Issue Conditions** Accumulator free at issue

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to addend register |
| CP 1 | *d* field latched in addend register<br>Transmit accumulator and addend register contents to shifter |
| CP 2 | First CP in shifter<br>Release addend register |
| CP 3 | Second CP in shifter<br>Transmit result to accumulator<br>Release accumulator |
| CP 4 | Latch result in accumulator |

**Description** Instruction 007 shifts the contents of the accumulator and the carry flag to the left in a circular mode. Bits $2^0$ through $2^4$ of the *d* field provide the shift count; bits $2^5$ through $2^8$ are not used. The carry flag is treated as bit $2^{16}$ of the accumulator. No shift is performed if the shift count is 0.



Instruction 010

| Machine Instruction | ELAN | Description |
|---|---|---|
| 010 | A = *d* | Transmit *d* to A |

**Issue Conditions** Accumulator free at issue

**Instruction Timing**

CP 0 Issue
Transmit *d* field to accumulator
Release accumulator

CP 1 Latch *d* field in accumulator

**Description** Instruction 010 transmits the *d* field bits $2^0$ through $2^8$ to the accumulator. Accumulator bits $2^9$ through $2^{15}$ and the carry flag are cleared.

Instruction 011

| Machine Instruction | ELAN | Description |
|---|---|---|
| 011 | A = A & *d* | Logical product of A and *d* to A |

**Issue Conditions** Accumulator free at issue

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to accumulator<br>Suppress Clear Accumulator signal<br>Release accumulator |
| CP 1 | Latch results in accumulator |

**Description** Instruction 011 transmits the logical product of the accumulator and the *d* field to the accumulator. Accumulator bits $2^9$ through $2^{15}$ are not changed. The instruction clears the carry flag.

Instruction 012

| Machine Instruction | ELAN | Description |
|---|---|---|
| 012 | A=A + *d* | Add *d* to A |

**Issue Conditions** Accumulator free at issue

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to addend register |
| CP 1 | *d* field latched in addend register<br>Transmit accumulator and addend register contents to adder |
| CP 2 | First CP in adder<br>Release addend register |
| CP 3 | Second CP in adder<br>Transmit result to accumulator<br>Release accumulator |
| CP 4 | Latch result in accumulator |

**Description** Instruction 012 transmits the two's complement sum of the accumulator and the *d* field to the accumulator. The *d* field is treated as a 9-bit positive integer. The instruction complements the carry flag if a carry bit is propagated from the accumulator bit $2^{15}$.

Instruction 013

| Machine Instruction | ELAN | Description |
|---|---|---|
| 013 | A = A - *d* | Subtract *d* from A |

**Issue Conditions** Accumulator free at issue

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to addend register |
| CP 1 | *d* field latched in addend register<br>Transmit accumulator and addend register contents to adder |
| CP 2 | First CP in adder<br>Invert *d* field<br>Release addend register |
| CP 3 | Second CP in adder<br>Transmit result to accumulator<br>Release accumulator |
| CP 4 | Latch result in accumulator |

**Description** Instruction 013 transmits the two's complement difference of the accumulator and the *d* field to the accumulator. The *d* field is treated as a 9-bit positive integer. The instruction complements the carry flag if a carry bit is propagated from accumulator bit $2^{15}$.

Instruction 014

| Machine Instruction | ELAN | Description |
|---|---|---|
| 014 | A = *k* | Transmit *k* to A |

**Issue Conditions**

Accumulator free at issue
NIP register valid 1 CP after issue

**Instruction Timing**

CP 0 Wait until NIP register valid
Issue
Transmit *k* field to accumulator
Release accumulator

CP 1 Latch *k* field in accumulator
Issue pass (000) instruction

**Description**

Instruction 014 transmits the *k* field to the accumulator and clears the carry flag.

Instruction 015

| Machine Instruction | ELAN | Description |
|---|---|---|
| 015 | A = A & *k* | Logical product of A and *k* to A |

**Issue Conditions**

Accumulator free at issue
NIP register valid 1 CP after issue

**Instruction Timing**

CP 0 Wait until NIP register valid
Issue
Transmit *k* field to accumulator
Suppress Clear Accumulator signal
Release accumulator

CP 1 Latch *k* field & accumulator contents in accumulator
Pass (000) in CIP register issued

**Description**

Instruction 015 transmits the logical product of the accumulator and the *k* field to the accumulator. Accumulator bits $2^9$ through $2^{15}$ are not changed. The instruction clears the carry flag.

Instruction 016

| Machine Instruction | ELAN | Description |
|---|---|---|
| 016 | A=A + *k* | Add *k* to A |

**Issue Conditions**

Accumulator free at issue
NIP register valid 1 CP after issue

**Instruction Timing**

CP 0 Wait until NIP register valid
Issue
Transmit *k* field to addend register
Accumulator reserved

CP 1 Latch *k* field in addend register
Issue pass (000) instruction
Transmit accumulator and addend register contents to adder

CP 2 First CP in adder
Release addend register

CP 3 Second CP in adder
Transmit result to accumulator
Release accumulator

CP 4 Latch result in accumulator

**Description**

Instruction 016 transmits the two's complement sum of the accumulator and the *k* field to the accumulator. The instruction complements the carry flag if a carry bit is propagated from the accumulator bit $2^{15}$.

Instruction 017

| Machine Instruction | ELAN | Description |
|---|---|---|
| 017 | A = A - *k* | Subtract *k* from A |

**Issue Conditions**

Accumulator free at issue
NIP register valid 1 CP after issue

**Instruction Timing**

CP 0 Wait until NIP register valid
Issue
Transmit *k* field to addend register

CP 1 Latch *k* field in addend register
Issue pass (000) instruction
Transmit accumulator and addend register contents to adder

CP 2 First CP in adder
Invert *k* field
Release addend register

CP 3 Second CP in adder
Transmit result to accumulator
Release accumulator

CP 4 Latch result in accumulator

**Description**

Instruction 017 transmits the two's complement difference of the accumulator and the *k* field to the accumulator. The instruction complements the carry flag if a carry bit is propagated from the accumulator bit $2^{15}$.

Instruction 020

| Machine Instruction | ELAN | Description |
|---|---|---|
| 020 | A = *dd* | Transmit operand register *dd* to A |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | Latch RP register<br>Read operand register |
| CP 2 | Latch operand register *dd* in read-out register<br>Check operand register parity<br>Release accumulator |
| CP 3 | Latch operand register *dd* in accumulator |

**Description**

Instruction 020 transmits the contents of the operand register designated by the *d* field to the accumulator. This instruction clears the carry flag.

Instruction 021

| Machine Instruction | ELAN | Description |
|---|---|---|
| 021 | A = A & *dd* | Logical product of A and operand register *dd* to A |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register

**Instruction Timing**

CP 0 Issue
Transmit *d* field to RP register

CP 1 Latch RP register
Read operand register *dd*

CP 2 Latch operand register *dd* in read-out register
Check operand register parity
Release accumulator
Suppress Clear Accumulator signal

CP 3 Latch result in accumulator

**Description**

Instruction 021 transmits the logical product of the accumulator and the contents of the operand register designated by the *d* field to the accumulator. The instruction clears the carry flag.

Instruction 022

| Machine Instruction | ELAN | Description |
|---|---|---|
| 022 | A=A + *dd* | Add operand register *dd* to A |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | RP register latched<br>Read operand register *dd* |
| CP 2 | Latch operand register *dd* in read-out register<br>Check operand register parity |
| CP 3 | Latch operand register *dd* in addend register |
| CP 4 | First CP in adder |
| CP 5 | Second CP in adder<br>Transmit result to accumulator<br>Release accumulator |
| CP 6 | Latch result in accumulator |

**Description**

Instruction 022 transmits the two's complement sum of the accumulator and the operand register designated by the *d* field to the accumulator. The operand register contents are treated as a 16-bit positive integer. The instruction complements the carry flag if a carry bit is propagated from the accumulator bit $2^{15}$.

Instruction 023

| Machine Instruction | ELAN | Description |
|---|---|---|
| 023 | A = A- *dd* | Subtract operand register *dd* from A |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | Latch RP register<br>Read operand register |
| CP 2 | Latch operand register *dd* in read-out register<br>Check operand register parity |
| CP 3 | Latch operand register *dd* in addend register |
| CP 4 | First CP in adder<br>Invert operand register *dd* for subtract |
| CP 5 | Second CP in adder<br>Transmit result to accumulator<br>Release accumulator |
| CP 6 | Latch result in accumulator |

**Description**

Instruction 023 transmits the two's complement difference of the accumulator and the operand register designated by the *d* field to the accumulator. The operand register contents are treated as a 16-bit positive integer. The instruction complements the carry flag if a carry bit is propagated from the accumulator bit $2^{15}$.

Instruction 024

| Machine Instruction | ELAN | Description |
|---|---|---|
| 024 | *dd* = A | Transmit A to operand register *dd* |

**Issue Conditions**

Accumulator free at issue
No delayed write of operand register

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register<br>Transmit accumulator contents to operand write register<br>Generate parity<br>Release accumulator |
| CP 1 | Enter operand write register contents into operand register |
| CP 2 | Result in operand register *dd* |

**Description**

Instruction 024 transmits the accumulator contents to the operand register designated by the *d* field.

Instruction 025

| Machine Instruction | ELAN | Description |
|---|---|---|
| 025 | *dd* = A + *dd* | Add operand register *dd* to A, result to operand register *dd* |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP and DP registers |
| CP 1 | Latch RP and DP registers<br>Read operand register |
| CP 2 | Latch operand register *dd* in read-out register<br>Check operand register parity |
| CP 3 | Latch operand register *dd* in addend register |
| CP 4 | First CP in adder |
| CP 5 | Second CP in adder<br>Transmit result to accumulator<br>Release accumulator |
| CP 6 | Latch result in accumulator<br>Transmit DP register contents to RP register<br>Generate parity<br>Transmit accumulator contents to operand write register |
| CP 7 | Transmit operand write register contents into operand register |
| CP 8 | Result in operand register *dd* |

**Description**

Instruction 025 computes the two's complement sum of the accumulator and the operand register designated by the *d* field. The result is transmitted to the accumulator and to the operand register designated by the *d* field. The instruction complements the carry flag if a carry bit is propagated from accumulator bit $2^{15}$.

Instruction 026

| Machine Instruction | ELAN | Description |
|---|---|---|
| 026 | *dd* = *dd* + 1 | Transmit operand register *dd* to A, add 1, result to operand register *dd* |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register

**Instruction Timing**

CP 0 Issue
Transmit *d* field to RP and DP registers

CP 1 Latch RP and DP registers
Read operand register

CP 2 Latch operand register *dd* in read-out register
Check operand register parity

CP 3 Latch operand register in accumulator
Latch constant 1 in addend register
Clear carry flag

CP 4 First CP in adder

CP 5 Second CP in adder
Transmit result to accumulator
Release accumulator

CP 6 Latch result in accumulator
Transmit DP register contents to RP register
Generate parity
Transmit accumulator contents to operand write register with parity

CP 7 Transmit operand write register to operand register *dd*

CP 8 Result in operand register *dd*

**Description**

Instruction 026 adds 1 to the contents of the operand register designated by the *d* field. The result is transmitted to the accumulator and to the operand register designated by the *d* field. The instruction complements the carry flag if a carry bit is propagated from accumulator bit $2^{15}$.

Instruction 027

| Machine Instruction | ELAN | Description |
|---|---|---|
| 027 | *dd* = *dd* - 1 | Transmit operand register *dd* to A, subtract 1, result to operand register *dd* |

### Issue Conditions

Accumulator free 1 CP after issue
No delayed write of operand register

### Instruction Timing

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP and DP registers |
| CP 1 | Latch RP and DP registers<br>Read operand register |
| CP 2 | Latch operand register *dd* in read-out register<br>Check operand register parity |
| CP 3 | Latch operand register in accumulator<br>Latch constant 1 in addend register<br>Clear carry flag |
| CP 4 | First CP in adder<br>Invert addend register |
| CP 5 | Second CP in adder<br>Transmit result to accumulator<br>Release accumulator |
| CP 6 | Latch result in accumulator<br>Transmit DP register contents to RP register<br>Generate parity<br>Transmit accumulator contents to operand write register with parity |
| CP 7 | Transmit operand write register contents into operand register *dd* |
| CP 8 | Result in operand register *dd* |

### Description

Instruction 027 subtracts 1 from the contents of the operand register designated by the *d* field. The result is transmitted to the accumulator and to the operand register designated by the *d* field. The instruction complements the carry flag if a carry bit is propagated from accumulator bit $2^{15}$.

Instruction 030

| Machine Instruction | ELAN | Description |
|---|---|---|
| 030 | A = [*dd*] | Transmit contents of memory addressed by operand register *dd* to A |

**Issue Conditions**

Accumulator free 3 CPs after issue
Accumulator not reserved by I/O or local memory write functions
No delayed write of operand register

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | Latch RP register<br>Read operand register |
| CP 2 | Latch operand register *dd* in read-out register<br>Check operand register parity<br>Request local memory read reference |
| CP 3 | Latch operand register *dd* in MA register |
| CP 4 | - - |
| CP 5 + N | Latch acknowledge<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs) |
| CP 6 + N | Acknowledge + 1 |
| CP 7 + N | Acknowledge + 2 |
| CP 8 + N | Acknowledge + 3 |
| CP 9 + N | Acknowledge + 4 |
| CP 10 + N | Acknowledge + 5<br>Release accumulator |
| CP 11 + N | Memory contents addressed by operand register *dd* in accumulator |

**Description**

Instruction 030 transmits the contents of a local memory location to the accumulator. The operand register specified by the *d* field supplies the local memory address. The instruction clears the carry flag.

Instruction 031

| Machine Instruction | ELAN | Description |
|---|---|---|
| 031 | A = A & [*dd*] | Logical product of A and contents of memory addressed by operand register *dd*, result to A |

**Issue Conditions**

Accumulator free 3 CPs after issue
Accumulator not reserved by I/O or local memory write functions
No delayed write of operand register

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | Latch RP register<br>Read operand register |
| CP 2 | Latch operand register *dd* in read-out register<br>Check operand register parity<br>Request local memory read reference |
| CP 3 | Latch operand register *dd* in MA register |
| CP 4 | -- |
| CP 5 + N | Acknowledge latched<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs) |
| CP 6 + N | Acknowledge + 1 |
| CP 7 + N | Acknowledge + 2 |
| CP 8 + N | Acknowledge + 3 |
| CP 9 + N | Acknowledge + 4 |
| CP 10 + N | Acknowledge + 5<br>Suppress Clear Accumulator Signal<br>Release accumulator |
| CP 11 + N | Latch result in accumulator |

**Description**

Instruction 031 transmits the logical product of the accumulator and a local memory location to the accumulator. The operand register specified by the *d* field provides the local memory address. The instruction clears the carry flag.

Instruction 032

| Machine Instruction | ELAN | Description |
|---|---|---|
| 032 | A = A + [*dd*] | Add contents of memory addressed by operand register *dd* to A |

**Issue Conditions**

Accumulator free 3 CPs after issue
Accumulator not reserved by I/O or local memory write functions
No delayed write of operand register

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | Latch RP register<br>Read operand register |
| CP 2 | Latch operand register *dd* in read-out register<br>Check operand register parity<br>Request local memory read reference |
| CP 3 | Latch operand register *dd* in MA register |
| CP 4 | - - |
| CP 5 + N | Latch acknowledge<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs) |
| CP 6 + N | Acknowledge + 1 |
| CP 7 + N | Acknowledge + 2 |
| CP 8 + N | Acknowledge + 3 |
| CP 9 + N | Acknowledge + 4 |
| CP 10 + N | Acknowledge + 5 |
| CP 11 + N | Latch contents of memory location *dd* in addend register<br>Transmit accumulator and addend register contents to adder |
| CP 12 + N | First CP in adder |
| CP 13 + N | Second CP in adder<br>Transmit results to accumulator<br>Release accumulator |
| CP 14 + N | Latch results in accumulator |

**Description**

Instruction 032 transmits the two's complement sum of the accumulator and a local memory location to the accumulator. The operand register specified by the *d* field provides the local memory address. The instruction complements the carry flag if a carry bit is propagated from accumulator bit $2^{15}$.

Instruction 033

| Machine Instruction | ELAN | Description |
|---|---|---|
| 033 | A = A - [*dd*] | Subtract contents of memory addressed by operand register *dd* from A |

**Issue Conditions**

Accumulator free 3 CPs after issue
Accumulator not reserved by I/O or local memory write functions
No delayed write of operand register

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | Latch RP register<br>Read operand register |
| CP 2 | Latch operand register *dd* in read-out register<br>Check operand register parity<br>Request local memory read reference |
| CP 3 | Latch operand register *dd* in MA register |
| CP 4 | - - |
| CP 5 + N | Acknowledge latched<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs) |
| CP 6 + N | Acknowledge + 1 |
| CP 7 + N | Acknowledge + 2 |
| CP 8 + N | Acknowledge + 3 |
| CP 9 + N | Acknowledge + 4 |
| CP 10 + N | Acknowledge + 5 |
| CP 11 + N | Latch contents of memory location *dd* in addend register<br>Transmit accumulator and addend register contents to adder |
| CP 12 + N | First CP in adder<br>Invert addend register |
| CP 13 + N | Second CP in adder<br>Transmit results to accumulator<br>Release accumulator |
| CP 14 + N | Latch results in accumulator |

**Description**

Instruction 033 transmits the two's complement difference of the accumulator and a local memory location to the accumulator. The operand register specified by the *d* field provides the local memory address. The instruction complements the carry flag if a carry bit is propagated from accumulator bit $2^{15}$.

Instruction 034

| Machine Instruction | ELAN | Description |
|---|---|---|
| 034 | [*dd*] = A | Transmit A to memory addressed by operand register *dd* |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | Latch RP register<br>Read operand register |
| CP 2 | Latch operand register *dd* in read-out register<br>Check operand register parity<br>Request local memory write reference |
| CP 3 | Latch operand register *dd* in MA register |
| CP 4 | - - |
| CP 5 + N | Latch acknowledge<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs)<br>Release accumulator |
| CP 6 + N | Memory write reference cycle CP 0 |
| CP 7 + N | Memory write reference cycle CP 1 |
| CP 8 + N | Memory write reference cycle CP 2 |

**Description**

Instruction 034 transmits the accumulator contents to a local memory address location. The operand register designated by the *d* field provides the local memory address.

Instruction 035

| Machine Instruction | ELAN | Description |
|---|---|---|
| 035 | [*dd*] = A + [*dd*] | Add memory addressed by operand register *dd* to A, result to same memory location |

**Issue Conditions**

Accumulator free 3 CPs after issue
Accumulator not reserved by I/O or local memory write functions
No delayed write of operand register

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | Latch RP register<br>Read operand register |
| CP 2 | Latch operand register *dd* in read-out register<br>Check operand register parity<br>Request local memory read reference |
| CP 3 | Latch operand register *dd* in MA register |
| CP 4 | - - |
| CP 5 + N | Latch acknowledge<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs) |
| CP 6 + N | Acknowledge + 1 |
| CP 7 + N | Acknowledge + 2 |
| CP 8 + N | Acknowledge + 3 |
| CP 9 + N | Acknowledge + 4 |
| CP 10 + N | Acknowledge + 5 |
| CP 11 + N | Latch contents of memory location *dd* in addend register<br>Transmit accumulator and addend register contents to adder |
| CP 12 + N | First CP in adder |
| CP 13 + N | Second CP in adder<br>Transmit results to accumulator<br>Request local memory write reference |

**Instruction Timing (continued)**

| | |
|---|---|
| CP 14 + N | Latch result in accumulator<br>Latch operand register *dd* in MA register |
| CP 15 + N | - - |
| CP 16 + N + M | Latch acknowledge<br>(M can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs)<br>Release accumulator |
| CP 17 + N + M | Memory write reference cycle CP 0 |
| CP 18 + N + M | Memory write reference cycle CP 1 |
| CP 19 + N + M | Memory write reference cycle CP 2 |

**Description**

Instruction 035 computes the two's complement sum of the accumulator and a local memory location. The operand register designated by the *d* field provides the local memory address. The result is transmitted to the accumulator and to the same local memory location. The instruction complements the carry flag if a carry bit is propagated from accumulator bit $2^{15}$.

Instruction 036

| Machine Instruction | ELAN | Description |
|---|---|---|
| 036 | [*dd*] = [*dd*] + 1 | Transmit memory addressed by operand register *dd* to A, add 1, result to same memory location |

**Issue Conditions**

Accumulator free 3 CPs after issue
Accumulator not reserved by I/O or local memory write functions
No delayed write of operand register

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | Latch RP register<br>Read operand register |
| CP 2 | Latch operand register *dd* in read-out register<br>Check operand register parity<br>Request local memory read reference |
| CP 3 | Latch operand register *dd* in MA register |
| CP 4 | - - |
| CP 5 + N | Latch acknowledge<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs) |
| CP 6 + N | Acknowledge + 1 |
| CP 7 + N | Acknowledge + 2 |
| CP 8 + N | Acknowledge + 3 |
| CP 9 + N | Acknowledge + 4 |
| CP 10 + N | Acknowledge + 5 |
| CP 11 + N | Latch contents of memory location *dd* in accumulator<br>Latch constant 1 in addend register<br>Transmit addend register contents and accumulator contents to adder |
| CP 12 + N | First CP in adder |

**Instruction Timing (continued)**

| | |
|---|---|
| CP 13 + N | Second CP in adder<br>Transmit results to accumulator<br>Request local memory write reference |
| CP 14 + N | Latch results in accumulator<br>Latch operand register *dd* in MA register |
| CP 15 + N | - - |
| CP 16 + N + M | Latch acknowledge<br>(M can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs)<br>Release accumulator |
| CP 17 + N + M | Memory write reference cycle CP 0 |
| CP 18 + N + M | Memory write reference cycle CP 1 |
| CP 19 + N + M | Memory write reference cycle CP 2 |

**Description**

Instruction 036 increments a local memory location by 1. The operand register designated by the *d* field provides the local memory address. The result is transmitted to the accumulator and to the local memory location. The instruction complements the carry flag if a carry bit is propagated from accumulator bit $2^{15}$.

Instruction 037

| Machine Instruction | ELAN | Description |
|---|---|---|
| 037 | [*dd*] = [*dd*] - 1 | Transmit memory addressed by operand register *dd* to A, subtract 1, result to same memory location |

**Issue Conditions**

Accumulator free 3 CPs after issue
Accumulator not reserved by I/O or local memory write functions
No delayed write of operand register

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | Latch RP register<br>Read operand register |
| CP 2 | Latch operand register *dd* in read-out register<br>Check operand register parity<br>Request local memory read reference |
| CP 3 | Latch operand register *dd* in MA register |
| CP 4 | - - |
| CP 5 + N | Latch acknowledge<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs) |
| CP 6 + N | Acknowledge + 1 |
| CP 7 + N | Acknowledge + 2 |
| CP 8 + N | Acknowledge + 3 |
| CP 9 + N | Acknowledge + 4 |
| CP 10 + N | Acknowledge + 5 |
| CP 11 + N | Latch contents of memory location *dd* in accumulator<br>Latch constant 1 in addend register<br>Transmit accumulator and addend register contents to adder |
| CP 12 + N | First CP in adder<br>Invert addend register |

**Instruction Timing (continued)**

| | |
|---|---|
| CP 13 + N | Second CP in adder<br>Transmit results to accumulator<br>Request local memory write reference |
| CP 14 + N | Latch result in accumulator<br>Latch operand register *dd* in MA register |
| CP 15 + N | - - |
| CP 16 + N + M | Latch acknowledge<br>(M can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs)<br>Release accumulator |
| CP 17 + N + M | Memory write reference cycle CP 0 |
| CP 18 + N + M | Memory write reference cycle CP 1 |
| CP 19 + N + M | Memory write reference cycle CP 2 |

**Description**

Instruction 037 decrements a local memory location by 1. The operand register designated by the *d* field provides the local memory address. The result is transmitted to the accumulator and to the local memory location. The instruction complements the carry flag if a carry bit is propagated from accumulator bit $2^{15}$.

Instruction 040

| Machine Instruction | ELAN | Description |
|---|---|---|
| 040 | C = 1, iod = = DN | Set carry equal to 1 if channel *d* done |

**Issue Conditions** Accumulator free 2 CPs after issue

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to internal I/O channels |
| CP 1 | -- |
| CP 2 | -- |
| CP 3 | Release accumulator |
| CP 4 | If channel *d* done flag set, then set carry flag |

**Description** Instruction 040 sets the carry flag to the same state as the done flag of the channel specified by the *d* field.

**Note:** For channels 10 through 37, when a channel adapter is present, there must be a delay of at least 16 CPs between any channel function that affects the done flag and an 040 instruction. If a local memory transfer is in progress, there must be a delay of 25 CPs.

Instruction 041

| Machine Instruction | ELAN | Description |
|---|---|---|
| 041 | C = 1, iod = = BZ | Set carry equal to 1 if channel *d* busy |

**Issue Conditions** Accumulator free 2 CPs after issue

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to internal I/O channels |
| CP 1 | - - |
| CP 2 | - - |
| CP 3 | Release accumulator |
| CP 4 | If channel *d* busy flag set, then set carry flag |

**Description** Instruction 041 sets the carry flag to the same state as the busy flag of the channel specified by the *d* field.

**Note:** For channels 10 through 37, when a channel adapter is present, there must be a delay of at least 16 CPs between any channel function that affects the busy flag and an 041 instruction. If a local memory transfer is in progress, there must be a delay of 25 CPs.

Instruction 042

| Machine Instruction | ELAN | Description |
|---|---|---|
| 042 | C = 1, IOB = = DN | Set carry equal to 1 if channel B done |

**Issue Conditions**

Accumulator free 2 CPs after issue
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to internal I/O channels |
| CP 1 | - - |
| CP 2 | - - |
| CP 3 | Release accumulator |
| CP 4 | If channel B done flag set, then set carry flag |

**Description**

Instruction 042 sets the carry flag to the same state as the done flag of the channel specified by the B register.

**Note:** For channels 10 through 37, when a channel adapter is present, there must be a delay of at least 16 CPs between any channel function that affects the done flag and an 042 instruction. If a local memory transfer is in progress, there must be a delay of 25 CPs.

Instruction 043

| Machine Instruction | ELAN | Description |
|---|---|---|
| 043 | C = 1, IOB = = BZ | Set carry equal to 1 if channel B busy |

**Issue Conditions**

Accumulator free 2 CPs after issue
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to internal I/O channels |
| CP 1 | -- |
| CP 2 | -- |
| CP 3 | Release accumulator |
| CP 4 | If channel B busy flag set, then set carry flag |

**Description**

Instruction 043 sets the carry flag to the same state as the busy flag of the channel specified by the B register.

**Note:** For channels 10 through 37, when a channel adapter is present, there must be a delay of at least 16 CPs between any channel function that affects the busy flag and an 043 instruction. If a local memory transfer is in progress, there must be a delay of 25 CPs.

Instruction 044

| Machine Instruction | ELAN | Description |
|---|---|---|
| 044 | A = A >! B | Right shift C and A by B places, end off |

**Issue Conditions**

Accumulator free at issue
B register not reserved

**Instruction Timing**

CP 0 Issue
Transmit B register contents to addend register

CP 1 B register contents latched in addend register
Transmit accumulator and addend register contents to shifter

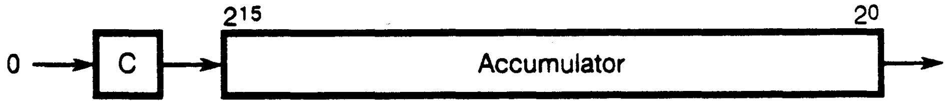
CP 2 First CP in shifter
Invert addend register
Release addend register

CP 3 Second CP in shifter
Transmit results to accumulator
Release accumulator

CP 4 Latch result in accumulator

**Description**

Instruction 044 shifts the contents of the accumulator and the carry flag to the right. Bits $2^0$ through $2^4$ of the B register provide the shift count; bits $2^5$ through $2^8$ are not used. The carry flag is treated as bit $2^{16}$ of the accumulator. The shift instruction enters 0's in the carry flag and propagates them to the right. No shift is performed if the shift count is 0. A shift count greater than 16 clears the accumulator and carry flag.



Instruction 045

| Machine Instruction | ELAN | Description |
|---|---|---|
| 045 | A = A <! B | Left shift C and A by B places, end off |

**Issue Conditions**

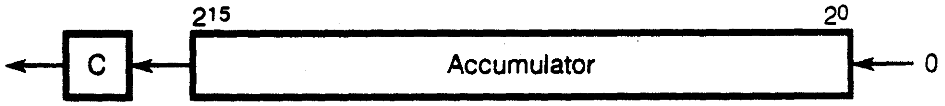
Accumulator free at issue
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to operand register |
| CP 1 | Latch B register contents in addend register<br>Transmit accumulator and addend register contents to shifter |
| CP 2 | First CP in shifter<br>Release addend register |
| CP 3 | Second CP in shifter<br>Transmit results to accumulator<br>Release accumulator |
| CP 4 | Latch result in accumulator |

**Description**

Instruction 045 shifts the contents of the accumulator and the carry flag to the left. Bits $2^0$ through $2^4$ of the B register provide the shift count; bits $2^5$ through $2^8$ are not used. The carry flag is treated as bit $2^{16}$ of the accumulator. The shift instruction enters 0's in accumulator bit $2^0$ and propagates them to the left. No shift is performed if the shift count is 0.



Instruction 046

| Machine Instruction | ELAN | Description |
|---|---|---|
| 046 | A = A >> B | Right shift C and A by B places, circular |

**Issue Conditions** Accumulator free at issue
B register not reserved

**Instruction Timing**

CP 0 Issue
Transmit B register contents to addend register

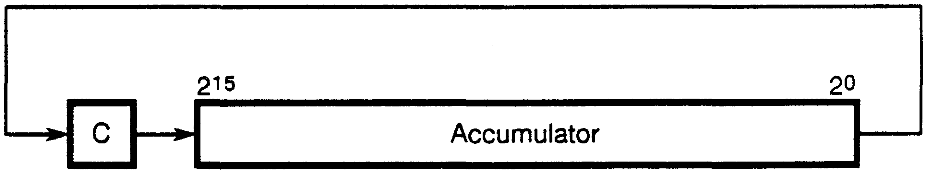
CP 1 Latch B register contents in addend register
Transmit accumulator and addend register contents to shifter

CP 2 First CP in shifter
Invert addend register
Release addend register

CP 3 Second CP in shifter
Transmit results to accumulator
Release accumulator

CP 4 Latch result in accumulator

**Description** Instruction 046 shifts the contents of the accumulator and the carry flag to the right in a circular mode. Bits $2^0$ through $2^4$ of the B register provide the shift count; bits $2^5$ through $2^8$ are not used. The carry flag is treated as bit $2^{16}$ of the accumulator. No shift is performed if the shift count is 0.



Instruction 047

| Machine Instruction | ELAN | Description |
|---|---|---|
| 047 | A = A << B | Left shift C and A by B places, circular |

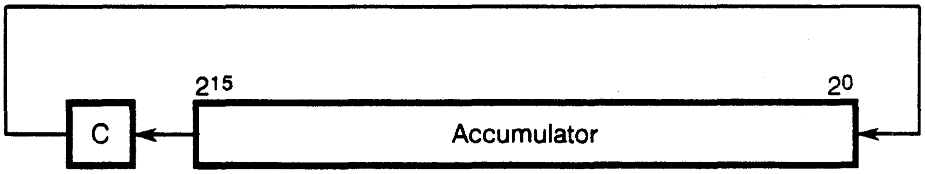
**Issue Conditions** Accumulator free at issue
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to addend register |
| CP 1 | Latch B register contents in addend register<br>Transmit accumulator and addend register contents to shifter |
| CP 2 | First CP in shifter<br>Release addend register |
| CP 3 | Second CP in shifter<br>Transmit results to accumulator<br>Release accumulator |
| CP 4 | Latch result in accumulator |

**Description**

Instruction 047 shifts the contents of the accumulator and the carry flag to the left in a circular mode. Bits $2^0$ through $2^4$ of the B register provide the shift count; bits $2^5$ through $2^8$ are not used. The carry flag is treated as bit $2^{16}$ of the accumulator. No shift is performed if the shift count is 0.



Instruction 050

| Machine Instruction | ELAN | Description |
|---|---|---|
| 050 | A = B | Transmit B to A |

**Issue Conditions**

Accumulator free at issue
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to accumulator<br>Release accumulator |
| CP 1 | Latch B register contents in accumulator |

**Description**

Instruction 050 transmits the B-register contents to bits $2^0$ through $2^8$. Accumulator bits $2^9$ through $2^{15}$ and the carry bit are cleared.

Instruction 051

| Machine Instruction | ELAN | Description |
|---|---|---|
| 051 | A = A & B | Logical product of A and B to A |

**Issue Conditions**

Accumulator free at issue
B register not reserved

**Instruction Timing**

CP 0 Issue
Transmit B register contents to accumulator
Suppress Clear Accumulator signal
Release accumulator

CP 1 Latch result in accumulator

**Description**

Instruction 051 transmits the logical product of the accumulator and the B register to the accumulator. Accumulator bits $2^9$ through $2^{15}$ are not changed. The instruction clears the carry flag.

Instruction 052

| Machine Instruction | ELAN | Description |
|---|---|---|
| 052 | A = A + B | Add B to A |

**Issue Conditions**

Accumulator free at issue
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to addend register |
| CP 1 | Latch B register contents in addend register<br>Transmit accumulator and addend register contents to adder |
| CP 2 | First CP in adder<br>Release addend register |
| CP 3 | Second CP in adder<br>Transmit results to accumulator<br>Release accumulator |
| CP 4 | Latch result in accumulator |

**Description**

Instruction 052 transmits the two's complement sum of the accumulator and the B register to the accumulator. The B register is treated as a 9-bit positive integer. The instruction complements the carry flag if a carry bit is propagated from the accumulator bit $2^{15}$.

Instruction 053

| Machine Instruction | ELAN | Description |
|---|---|---|
| 053 | A = A - B | Subtract B from A |

**Issue Conditions**

Accumulator free at issue
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to addend register |
| CP 1 | Latch B register contents in addend register<br>Transmit accumulator and addend register contents to adder |
| CP 2 | First CP in adder<br>Invert addend register<br>Release addend register |
| CP 3 | Second CP in adder<br>Transmit results to accumulator<br>Release accumulator |
| CP 4 | Latch result in accumulator |

**Description**

Instruction 053 transmits the two's complement difference of the accumulator and the B register to the accumulator. The B register is treated as a 9-bit positive integer. The instruction complements the carry flag if a carry bit is propagated from the accumulator bit $2^{15}$.

Instruction 054

| Machine Instruction | ELAN | Description |
|---|---|---|
| 054 | B = A | Transmit A to B |

**Issue Conditions** Accumulator free at issue
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit accumulator contents to B register |
| CP 1 | Latch accumulator contents in B register |

**Description** Instruction 054 transmits accumulator bits $2^0$ through $2^8$ to the B register.

Instruction 055

| Machine Instruction | ELAN | Description |
|---|---|---|
| 055 | B = A + B | Add B to A, result to B |

**Issue Conditions**

Accumulator free at issue
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to addend register<br>Reserve B register |
| CP 1 | Transmit accumulator and addend register contents to adder |
| CP 2 | First CP in adder |
| CP 3 | Second CP in adder<br>Transmit result to accumulator<br>Release accumulator |
| CP 4 | Latch result in accumulator<br>Transmit accumulator contents to B register<br>Release B register |
| CP 5 | Latch result in B register |

**Description**

Instruction 055 computes the two's complement sum of the accumulator and the B register. The result is transmitted to the accumulator and to the B register. The instruction complements the carry flag if a carry bit is propagated from accumulator bit $2^{15}$.

Instruction 056

| Machine Instruction | ELAN | Description |
|---|---|---|
| 056 | B = B + 1 | Transmit B to A, add 1, result to B |

**Issue Conditions**

Accumulator free at issue
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to accumulator<br>Enter constant 1 into addend register<br>Reserve B register |
| CP 1 | Transmit accumulator and addend register contents to adder |
| CP 2 | First CP in adder |
| CP 3 | Second CP in adder<br>Transmit result to accumulator<br>Release accumulator |
| CP 4 | Latch result in accumulator<br>Transmit accumulator contents to B register<br>Release B register |
| CP 5 | Latch result in B register |

**Description**

Instruction 056 adds 1 to the contents of the B register. The result is transmitted to the accumulator and to the B register. Accumulator bits $2^{10}$ through $2^{15}$ and the carry bit are cleared.

Instruction 057

| Machine Instruction | ELAN | Description |
|---|---|---|
| 057 | B = B - 1 | Transmit B to A, subtract 1, result to B |

**Issue Conditions**

Accumulator free at issue
B register not reserved

**Instruction Timing**

CP 0 Issue
Transmit B register contents to accumulator
Enter constant 1 into addend register
Reserve B register

CP 1 Transmit accumulator and addend register contents to adder

CP 2 First CP in adder
Invert addend register

CP 3 Second CP in adder
Transmit result to accumulator
Release accumulator

CP 4 Latch result in accumulator
Transmit accumulator contents to B register
Release B register

CP 5 Result in B register

**Description**

Instruction 057 subtracts 1 from the contents of the B register. The result is transmitted to the accumulator and to the B register. The instruction complements the carry flag if a carry bit is propagated from accumulator bit $2^{15}$.

Instruction 060

| Machine Instruction | ELAN | Description |
|---|---|---|
| 060 | A = [B] | Transmit operand register B to A |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to RP register |
| CP 1 | Latch RP register contents<br>Read operand register |
| CP 2 | Latch operand register contents in read-out register<br>Check operand register parity<br>Release accumulator |
| CP 3 | Latch operand register contents in accumulator |

**Description**

Instruction 060 transmits the contents of the operand register designated by the B register into the accumulator. It then clears the carry flag.

**Note:** Because there are only $200_8$ operand registers, B register bits $2^7$ and $2^8$ are ignored.

Instruction 061

| Machine Instruction | ELAN | Description |
|---|---|---|
| 061 | A = A & [B] | Logical product of A and operand register B to A |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register
B register not reserved

**Instruction Timing**

CP 0 Issue
Transmit B register contents to RP register

CP 1 Latch RP register contents
Read operand register

CP 2 Latch operand register contents in read-out register
Check operand register parity
Release accumulator
Suppress clear-accumulator function

CP 3 Latch result in accumulator

**Description**

Instruction 061 transmits the logical product of the accumulator and operand register B to the accumulator. Accumulator bits $2^9$ through $2^{15}$ are not changed. It then clears the carry flag.

**Note:** Because there are only $200_8$ operand registers, B register bits $2^7$ and $2^8$ are ignored.

Instruction 062

| Machine Instruction | ELAN | Description |
|---|---|---|
| 062 | A = A + [B] | Add operand register B to A |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to RP register |
| CP 1 | Latch RP register contents<br>Read operand register |
| CP 2 | Latch operand register contents in read-out register<br>Check operand register parity |
| CP 3 | Latch operand register contents in addend register |
| CP 4 | First CP in adder |
| CP 5 | Second CP in adder<br>Transmit result to accumulator<br>Release accumulator |
| CP 6 | Latch result in accumulator |

**Description**

Instruction 062 transmits the two's complement sum of the accumulator and operand register B to the accumulator. The instruction complements the carry flag if a carry bit is propagated from the accumulator bit $2^{15}$.

**Note:** Because there are only $200_8$ operand registers, B register bits $2^7$ and $2^8$ are ignored.

Instruction 063

| Machine Instruction | ELAN | Description |
|---|---|---|
| 063 | A = A - [B] | Subtract operand register B from A |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to RP register |
| CP 1 | Latch RP register contents<br>Read operand register |
| CP 2 | Latch operand register contents in read-out register<br>Check operand register parity |
| CP 3 | Latch operand register contents in addend register |
| CP 4 | First CP in adder<br>Invert B register for subtract |
| CP 5 | Second CP in adder<br>Transmit result to accumulator<br>Release accumulator |
| CP 6 | Latch result in accumulator |

**Description**

Instruction 063 transmits the two's complement difference of the accumulator and operand register B to the accumulator. The instruction complements the carry flag if a carry bit is propagated from the accumulator bit $2^{15}$.

**Note:** Because there are only $200_8$ operand registers, B register bits $2^7$ and $2^8$ are ignored.

Instruction 064

| Machine Instruction | ELAN | Description |
|---|---|---|
| 064 | [B] = A | Transmit A to operand register B |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to RP register<br>Generate parity<br>Transmit accumulator contents to operand write register<br>Release accumulator |
| CP 1 | Transmit operand write register contents to operand register |
| CP 2 | Write operand register |

**Description**

Instruction 064 transmits the accumulator contents to the operand register designated by the B register.

**Note:** Because there are only $200_8$ operand registers, B register bits $2^7$ and $2^8$ are ignored.

Instruction 065

| Machine Instruction | ELAN | Description |
|---|---|---|
| 065 | [B] = A + [B] | Add operand register B to A, result to operand register B |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to RP and DP registers |
| CP 1 | Latch RP and DP registers<br>Read operand register |
| CP 2 | Latch operand register contents in read-out register<br>Check operand register parity |
| CP 3 | Latch operand register contents in addend register |
| CP 4 | First CP in adder |
| CP 5 | Second CP in adder<br>Transmit result to accumulator<br>Release accumulator |
| CP 6 | Latch result in accumulator<br>Transmit DP register contents to RP register<br>Generate parity<br>Transmit accumulator contents to operand write register<br>Release DP register |
| CP 7 | Transmit operand write register contents to operand register |
| CP 8 | Write operand register |

**Description**

Instruction 065 computes the two's complement sum of the accumulator and the operand register designated by operand register B. The result is transmitted to the accumulator and to the operand register designated by the B register. The instruction complements the carry flag if a carry bit is propagated from accumulator bit $2^{15}$.

**Note:** Because there are only $200_8$ operand registers, B register bits $2^7$ and $2^8$ are ignored.

Instruction 066

| Machine Instruction | ELAN | Description |
|---|---|---|
| 066 | [B] = [B] + 1 | Transmit operand register B to A, add 1, result to operand register B |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to RP and DP registers |
| CP 1 | Latch RP and DP register contents<br>Read operand register |
| CP 2 | Latch operand register contents in read-out register<br>Check operand register parity |
| CP 3 | Latch operand register contents in accumulator<br>Latch constant 1 in addend register<br>Clear carry flag |
| CP 4 | First CP in adder |
| CP 5 | Second CP in adder<br>Transmit result to accumulator<br>Release accumulator |
| CP 6 | Latch result in accumulator<br>Transmit DP register contents to RP register<br>Transmit accumulator contents to operand write register<br>Release DP register |
| CP 7 | Transmit operand write register contents to operand register |
| CP 8 | Write operand register |

**Description**

Instruction 066 adds 1 to the contents of the operand register designated by the B register. The result is transmitted to the accumulator and to the operand register designated by the B register. The instruction complements the carry bit if a carry bit is propagated from accumulator bit $2^{15}$.

**Note:** Because there are only $200_8$ operand registers, B register bits $2^7$ and $2^8$ are ignored.

Instruction 067

| Machine Instruction | ELAN | Description |
|---|---|---|
| 067 | [B] = [B] - 1 | Transmit operand register B to A, subtract 1, result to operand register B |

**Issue Conditions**

Accumulator free 1 CP after issue
No delayed write of operand register
B register not reserved

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit B register contents to RP and DP registers |
| CP 1 | Latch RP and DP register contents<br>Read operand register |
| CP 2 | Latch operand register contents in read-out register<br>Check operand register parity |
| CP 3 | Latch operand register contents in accumulator<br>Latch constant 1 in addend register<br>Clear carry flag |
| CP 4 | First CP in adder<br>Invert addend register |
| CP 5 | Second CP in adder<br>Transmit result to accumulator<br>Release accumulator |
| CP 6 | Latch result in accumulator<br>Transmit DP register contents to RP register<br>Transmit accumulator contents to operand write register<br>Release DP register |
| CP 7 | Transmit operand write register contents to operand register |
| CP 8 | Write operand register |

**Description**

Instruction 067 subtracts 1 from the contents of the operand register designated by operand register B. The result is transmitted to the accumulator and to the operand register designated by operand register B. The instruction complements the carry bit if a carry bit is propagated from accumulator bit $2^{15}$.

**Note:** Because there are only $200_8$ operand registers, B register bits $2^7$ and $2^8$ are ignored.

Instruction 070

| Machine Instruction | ELAN | Description |
|---|---|---|
| 070 | P = p + *d* | Jump to p + *d* |

| | | |
|---|---|---|
| **Issue Conditions** | No adder/shifter path conflicts 2 CPs after issue<br>No adder/shifter path reservation for I/O functions | |
| **Instruction Timing** | CP 0 | Issue<br>Transmit P register contents to branch accumulator<br>Transmit *d* field to addend register |
| | CP 1 | Transmit branch accumulator and addend register contents to adder |
| | CP 2 | First CP in adder |
| | CP 3 | Second CP in adder<br>Transmit result to P register |
| If in-stack branch | CP 4 | Latch result in P register<br>Instruction stack pointer valid |
| | CP 5 | Instruction stack read pointer valid |
| | CP 6 | Latch next instruction in NIP register |
| | CP 7 | Latch next instruction in CIP register<br>Issue |
| If out-of-stack branch | CP 4 | Latch result in P register<br>Clear instruction stack<br>Request instruction fetch |
| | CP 5 + N | Fetch pointer valid<br>Acknowledge memory reference<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs) |
| | CP 6 + N | Latch acknowledge |
| | CP 7 + N | Local memory cycle CP 0 |
| | CP 8 + N | Local memory cycle CP 1 |
| | CP 9 + N | Local memory cycle CP 2 |

| | | |
|---|---|---|
| If out-of-stack branch (continued) | CP 10 + N | Latch local memory read data |
| | CP 11 + N | Correct local memory read data<br>Transmit corrected data to NIP register |
| | CP 12 + N | Latch next instruction in NIP register |
| | CP 13 + N | Latch next instruction in CIP register<br>Issue next instruction |

**Description**

Instruction 070 terminates the current program sequence and begins a new sequence. The initial address for the new sequence equals the address of the current instruction plus the *d* field. The operation treats the *d* field as a 9-bit positive integer and performs addition in a 16-bit, two's complement mode. The operation does not alter the accumulator or the carry flag.

Instruction 071

| Machine Instruction | ELAN | Description |
|---|---|---|
| 071 | P = p - *d* | Jump to p - *d* |

**Issue Conditions** No adder/shifter path conflicts 2 CPs after issue
No adder/shifter path reservation for I/O functions

| | | |
|---|---|---|
| **Instruction Timing** | CP 0 | Issue<br>Transmit P register contents to branch accumulator<br>Transmit *d* field to addend register |
| | CP 1 | Transmit branch accumulator and addend register contents to adder |
| | CP 2 | First CP in adder<br>Invert addend register |
| | CP 3 | Second CP in adder<br>Transmit result to P register |
| If in-stack branch | CP 4 | Latch result in P register<br>Instruction stack pointer valid |
| | CP 5 | Instruction stack read pointer valid |
| | CP 6 | Latch next instruction in NIP register |
| | CP 7 | Latch next instruction in CIP register<br>Issue |
| If out-of-stack branch | CP 4 | Latch result in P register<br>Clear instruction stack<br>Request instruction fetch |
| | CP 5 + N | Fetch pointer valid<br>Acknowledge memory reference<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs.) |
| | CP 6 + N | Latch acknowledge |
| | CP 7 + N | Local memory cycle CP 0 |
| | CP 8 + N | Local memory cycle CP 1 |

| | | |
|---|---|---|
| If out-of-stack branch (continued) | CP 9 + N | Local memory cycle CP 2 |
| | CP 10 + N | Latch local memory read data |
| | CP 11 + N | Correct local memory read data<br>Transmit corrected data to NIP register |
| | CP 12 + N | Latch next instruction in NIP register |
| | CP 13 + N | Latch next instruction in CIP register<br>Issue next instruction |

**Description**

Instruction 071 terminates the current program sequence and begins a new sequence. The initial address for the new sequence equals the address of the current instruction minus the *d* field. The operation performs subtraction in a 16-bit, two's complement mode, treating the *d* field as a 9-bit positive integer. The operation does not alter the accumulator or the carry flag.

Instruction 072

| Machine Instruction | ELAN | Description |
|---|---|---|
| 072 | R = p + *d* | Return jump to p + *d* |

| | | |
|---|---|---|
| **Issue Conditions** | No adder/shifter path conflicts 2 CPs after issue<br>No adder/shifter path reservation for I/O functions | |
| **Instruction Timing** | CP 0 | Issue<br>Transmit P register contents to branch accumulator<br>Transmit *d* field to addend register |
| | CP 1 | Transmit branch accumulator and addend register contents to adder<br>Increment P to P + 1 |
| | CP 2 | First CP in adder |
| | CP 3 | Second CP in adder<br>Transmit result to P register<br>Increment E pointer<br>Transmit P + 1 to exit stack |
| If in-stack branch | CP 4 | Latch result in P register<br>Instruction stack pointer valid |
| | CP 5 | Instruction stack read pointer valid |
| | CP 6 | Latch next instruction in NIP register |
| | CP 7 | Latch next instruction in CIP register<br>Issue |
| If out-of-stack branch | CP 4 | Latch result in P register<br>Clear instruction stack<br>Request instruction fetch |
| | CP 5 + N | Fetch pointer valid<br>Acknowledge memory reference<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs.) |
| | CP 6 + N | Latch acknowledge |
| | CP 7 + N | Local memory cycle CP 0 |
| | CP 8 + N | Local memory cycle CP 1 |

| | | |
|---|---|---|
| If out-of-stack branch (continued) | CP 9 + N | Local memory cycle CP 2 |
| | CP 10 + N | Latch local memory read data |
| | CP 11 + N | Correct local memory read data<br>Transmit corrected data to NIP register |
| | CP 12 + N | Latch next instruction in NIP register |
| | CP 13 + N | Latch next instruction in CIP register<br>Issue next instruction |

### Description

Instruction 072 suspends execution of the current program sequence and calls a subprogram for execution. The instruction increments the value of the E pointer by 1 and stores the address of the next instruction of current program sequence in the program exit stack. Then, the operation begins executing a new program sequence. The initial address for the new sequence equals the address of the current instruction plus the *d* field. The operation treats the *d* field as a 9-bit positive integer and adds in a 16-bit, two's complement mode. The instruction does not alter the accumulator or the carry flag.

The program exit stack overflow flag sets if the incremented E value is 30 ($36_8$).

Instruction 073

| Machine Instruction | ELAN | Description |
|---|---|---|
| 073 | R = p - *d* | Return jump to p - *d* |

**Issue Conditions** No adder/shifter path conflicts 2 CPs after issue
No adder/shifter path reservation for I/O functions

| | | |
|---|---|---|
| **Instruction Timing** | CP 0 | Issue<br>Transmit P register contents to branch accumulator<br>Transmit *d* field to addend register |
| | CP 1 | Transmit branch accumulator and addend register contents to adder<br>Advance P to P + 1 |
| | CP 2 | First CP in adder<br>Invert addend register |
| | CP 3 | Second CP in adder<br>Transmit result to P register<br>Increment E pointer<br>Transmit P + 1 to exit stack |
| If in-stack branch | CP 4 | Latch result in P register<br>Instruction stack pointer valid |
| | CP 5 | Instruction stack read pointer valid |
| | CP 6 | Latch next instruction in NIP register |
| | CP 7 | Latch next instruction in CIP register<br>Issue |
| If out-of-stack branch | CP 4 | Latch result in P register<br>Clear instruction stack<br>Instruction fetch request |
| | CP 5 + N | Fetch pointer valid<br>Acknowledge memory reference<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs.) |
| | CP 6 + N | Latch acknowledge |
| | CP 7 + N | Local memory cycle CP 0 |
| | CP 8 + N | Local memory cycle CP 1 |

| | | |
|---|---|---|
| If out-of-stack branch (continued) | CP 9 + N | Local memory cycle CP 2 |
| | CP 10 + N | Latch local memory read data |
| | CP 11 + N | Correct local memory read data<br>Transmit corrected data to NIP register |
| | CP 12 + N | Latch next instruction in NIP register |
| | CP 13 + N | Latch next instruction in CIP register<br>Issue next instruction |

**Description**

Instruction 073 suspends execution of the current program sequence and calls a subprogram for execution. The instruction increments the value of the E register by 1 and stores the address of the next sequential instruction of current program sequence in the program exit stack. Then, the instruction begins a new program sequence. The initial address for the new sequence equals the address of the current instruction minus the *d* field. The operation treats the d field as a 9-bit positive integer and performs subtraction in a 16-bit, two's complement mode. The instruction does not alter the accumulator or the carry flag.

**Note:** The program exit stack overflow flag sets if the incremented E value is 30 ($36_8$).

Instruction 074

| Machine Instruction | ELAN | Description |
|---|---|---|
| 074 | P = *dd* | Jump to address in operand register *dd* |

**Issue Conditions**

No adder/shifter path reservation for I/O functions
No delayed write of operand register
No local memory read cycle at acknowledge + 2

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | Latch RP register contents<br>Read operand register |
| CP 2 | Latch operand register in read-out register<br>Check operand register parity |
| CP 3 | Latch operand register *dd* in branch accumulator<br>Increment P register to P + 1<br>Transmit branch accumulator and addend register contents to adder |
| CP 4 | First CP in adder |
| CP 5 | Second CP in adder |
| CP 6 | Latch result in P register<br>Clear instruction stack<br>Request instruction fetch |
| CP 5 + N | Fetch pointer valid<br>Acknowledge memory reference<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs.) |
| CP 6 + N | Latch acknowledge |
| CP 7 + N | Local memory cycle CP 0 |
| CP 8 + N | Local memory cycle CP 1 |
| CP 9 + N | Local memory cycle CP 2 |
| CP 10 + N | Latch local memory read data |
| CP 11 + N | Correct local memory read data<br>Transmit corrected data to NIP register |

**Instruction Timing (continued)**

CP 12 + N Latch fetch data in NIP register

CP 13 + N Latch next instruction in CIP register
Issue next instruction

**Description**

Instruction 074 terminates the current program sequence and begins a new sequence. The operand register designated by the *d* field provides the initial address for the new sequence.

Instruction 075

| Machine Instruction | ELAN | Description |
|---|---|---|
| 075 | P = *dd* + *k* | Jump to sum of *k* and operand register *dd* |

**Issue Conditions** No adder/shifter path reservation for I/O functions
No delayed write of operand register
No local memory read cycle at acknowledge + 2
NIP register valid 1 CP after issue

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | Latch RP register contents<br>Read operand register |
| CP 2 | Latch operand register in read-out register<br>Check operand register parity<br>Transmit *k* field to addend register |
| CP 3 | Operand register *dd* latched in branch accumulator<br>Latch *k* field in addend register<br>Transmit branch accumulator and addend register contents to adder<br>Increment P register to P+1 |
| CP 4 | First CP in adder<br>Increment P register to P+2 |
| CP 5 | Second CP in adder |
| CP 6 | Latch result in P register<br>Clear instruction stack<br>Request instruction fetch |
| CP 5 + N | Fetch pointer valid<br>Acknowledge memory reference<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs.) |
| CP 6 + N | Latch acknowledge |
| CP 7 + N | Local memory cycle CP 0 |
| CP 8 + N | Local memory cycle CP 1 |
| CP 9 + N | Local memory cycle CP 2 |
| CP 10 + N | Latch local memory read data |

**Instruction Timing (continued)**

| | |
|---|---|
| CP 11 + N | Correct local memory read data<br>Transmit corrected data to NIP register |
| CP 12 + N | Latch fetch data in NIP register |
| CP 13 + N | Latch next instruction in CIP register<br>Issue next instruction |

**Description**

Instruction 075 terminates the current program sequence and begins a new sequence. The initial address for the new sequence equals the instruction *k* field plus the contents of the operand register specified by the *d* field. The operation performs addition in a 16-bit, two's complement mode. The operation does not alter the contents of the accumulator or the carry flag.

Instruction 076

| Machine Instruction | ELAN | Description |
|---|---|---|
| 076 | R = *dd* | Return jump to address in operand register *dd* |

**Issue Conditions**

No adder/shifter path reservation for I/O functions
No delayed write of operand register
No local memory read cycle at acknowledge + 2

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | Latch RP register contents<br>Read operand register |
| CP 2 | Latch operand register in read-out register<br>Check operand register parity |
| CP 3 | Latch operand register *dd* in branch accumulator<br>Transmit branch accumulator contents and addend register contents to adder<br>Increment P register to P + 1 |
| CP 4 | First CP in adder |
| CP 5 | Second CP in adder<br>Increment E pointer to E + 1<br>Transmit P +1 to exit stack |
| CP 6 | Latch result in P register<br>Clear instruction stack<br>Request instruction fetch |
| CP 5 + N | Fetch pointer valid<br>Acknowledge memory reference<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs.) |
| CP 6 + N | Latch acknowledge |
| CP 7 + N | Local memory cycle CP 0 |
| CP 8 + N | Local memory cycle CP 1 |
| CP 9 + N | Local memory cycle CP 2 |
| CP 10 + N | Latch local memory read data |
| CP 11 + N | Correct local memory read data<br>Transmit corrected data to NIP register |

## Instruction Timing (continued)

| | |
|---|---|
| CP 12 + N | Latch fetch data in NIP register |
| CP 13 + N | Latch next instruction in CIP register<br>Issue next instruction |

## Description

Instruction 076 suspends execution of the current program sequence and calls a subprogram for execution. The instruction increments the value of the E pointer by 1, stores the address of the next instruction of the current sequence in the program exit stack, and begins a new program sequence. The operand register specified by the *d* field provides the initial address for the new sequence.

**Note:** The program exit stack overflow flag sets if the incremented E value is 30 ($36_8$).

Instruction 077

| Machine Instruction | ELAN | Description |
|---|---|---|
| 077 | R = *dd* + *k* | Return jump to address sum of *k* and operand register *dd* |

**Issue Conditions**

No adder/shifter path reservation for I/O functions
No delayed write of operand register (DP register full)
No local memory read cycle at acknowledge + 2
NIP register valid 1 CP after issue

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field to RP register |
| CP 1 | Latch RP register contents<br>Read operand register |
| CP 2 | Latch operand register in read-out register<br>Check operand register parity<br>Transmit *k* field to addend register |
| CP 3 | Latch operand register *dd* in branch accumulator<br>Latch *k* field in addend register<br>Transmit branch accumulator and addend register contents to adder<br>Increment P register to P + 1 |
| CP 4 | First CP in adder<br>Increment P register to P +2 |
| CP 5 | Second CP in adder<br>Increment E pointer to E + 1<br>Transmit P +2 to exit stack |
| CP 6 | Latch result in P register<br>Clear instruction stack<br>Request instruction fetch |
| CP 5 + N | Fetch pointer valid<br>Acknowledge reference<br>(N can vary from 0 to 5 CPs depending on local memory timing and conflicts, but is typically not more than 2 CPs.) |
| CP 6 + N | Latch acknowledge |
| CP 7 + N | Local memory cycle CP 0 |
| CP 8 + N | Local memory cycle CP 1 |
| CP 9 + N | Local memory cycle CP 2 |

**Instruction Timing (continued)**

| | |
|---|---|
| CP 10 + N | Latch local memory read data |
| CP 11 + N | Correct local memory read data<br>Transmit corrected data to NIP register |
| CP 12 + N | Latch fetch data in NIP register |
| CP 13 + N | Latch next instruction in CIP register<br>Issue next instruction |

**Description**

Instruction 077 suspends the current program sequence and calls a subprogram for execution. The instruction increments the value of the E pointer by one and stores the address of the next instruction of the current sequence in the program exit stack. A new program sequence then begins. The initial address for the new sequence equals the *k* field plus the contents of operand register designated by the *d* field. Addition is performed in a 16-bit, two's complement mode. The operation does not alter the contents of the accumulator or the carry flag.

The program exit stack overflow flag sets if the incremented E value is 30 ($36_8$).

## Instructions 100 through 137

Instructions 100 through 137 are branch instructions that are used to jump to a new program sequence. Refer to Table 7-4 for a list of instructions 100 through 137.

Table 7-4. Instructions 100 through 137

| Machine Instruction | ELAN | Description |
|---|---|---|
| 100 | P = p + *d*, C = = 0 | Jump to p + *d* if carry = 0 |
| 101 | P = p + *d*, C != 0 | Jump to p + *d* if carry ≠ 0 |
| 102 | P = p + *d*, A = = 0 | Jump to p + *d* if A = 0 |
| 103 | P = p + *d*, A != 0 | Jump to p + *d* if A ≠ 0 |
| 104 | P = p - *d*, C = = 0 | Jump to p - *d* if carry = 0 |
| 105 | P = p - *d*, C != 0 | Jump to p - *d* if carry ≠ 0 |
| 106 | P = p - *d*, A = = 0 | Jump to p - *d* if A = 0 |
| 107 | P = p - *d*, A != 0 | Jump to p - *d* if A ≠ 0 |
| 110 | R = p + *d*, C = = 0 | Return jump to p + *d* if carry = 0 |
| 111 | R = p + *d*, C != 0 | Return jump to p + *d* if carry ≠ 0 |
| 112 | R = p + *d*, A = = 0 | Return jump to p + *d* if A = 0 |
| 113 | R = p + *d*, A != 0 | Return jump to p + *d* if A ≠ 0 |
| 114 | R = p - *d*, C = = 0 | Return jump to p - *d* if carry = 0 |
| 115 | R = p - *d*, C != 0 | Return jump to p - *d* if carry ≠ 0 |
| 116 | R = p - *d*, A = = 0 | Return jump to p - *d* if A = 0 |
| 117 | R = p - *d*, A != 0 | Return jump to p - *d* if A ≠ 0 |
| 120 | P = *dd*, C = = 0 | Jump to address in operand register *dd* if carry = 0 |
| 121 | P = *dd*, C != 0 | Jump to address in operand register *dd* if carry ≠ 0 |
| 122 | P = *dd*, A = = 0 | Jump to address in operand register *dd* if A= 0 |
| 123 | P = *dd*, A != 0 | Jump to address in operand register *dd* if A ≠ 0 |
| 124 | P = *dd* + *k*, C = = 0 | Jump to address in operand register *dd* + *k* if carry = 0 |
| 125 | P = *dd* + *k*, C != 0 | Jump to address in operand register *dd* + *k* if carry ≠ 0 |
| 126 | P = *dd* + *k*, A = = 0 | Jump to address in operand register *dd* + *k* if A = 0 |
| 127 | P = *dd* + *k*, A != 0 | Jump to address in operand register *dd* + *k* if A ≠ 0 |
| 130 | R = *dd*, C = = 0 | Return jump to address in operand register *dd* if carry = 0 |

Table 7-4. Instructions 100 through 137 (continued)

| Machine Instruction | ELAN | Description |
|---|---|---|
| 131 | R = *dd*, C != 0 | Return jump to address in operand register *dd* if carry ≠ 0 |
| 132 | R = *dd*, A = = 0 | Return jump to address in operand register *dd* if A = 0 |
| 133 | R = *dd*, A != 0 | Return jump to address in operand register *dd* if A ≠ 0 |
| 134 | R = *dd* + *k*, C = = 0 | Return jump to address in operand register *dd* + *k* if carry = 0 |
| 135 | R = *dd* + *k*, C != 0 | Return jump to address in operand register *dd* + *k* if carry ≠ 0 |
| 136 | R = *dd* + *k*, A = = 0 | Return jump to address in operand register *dd* + *k* if A = 0 |
| 137 | R = *dd* + *k*, A != 0 | Return jump to address in operand register *dd* + *k* if A ≠ 0 |

**Issue Conditions**

Each instruction has the same issue conditions as the corresponding unconditional branch instruction (instructions 070 through 077). A list of the branch instructions is provided in Table 7-5.

Table 7-5. Branch Instructions

| ELAN | Unconditional Branch Instruction | Conditional Branch Instructions | | | |
|---|---|---|---|---|---|
| | | C = 0 | C ≠ 0 | A = 0 | A ≠ 0 |
| P=p + *d* | 070 | 100 | 101 | 102 | 103 |
| P = p - *d* | 071 | 104 | 105 | 106 | 107 |
| R = p + *d* | 072 | 110 | 111 | 112 | 113 |
| R = p - *d* | 073 | 114 | 115 | 116 | 117 |
| P = *dd* | 074 | 120 | 121 | 122 | 123 |
| P = *dd* + *k* | 075 | 124 | 125 | 126 | 127 |
| R = *dd* | 076 | 130 | 131 | 132 | 133 |
| R = *dd* + *k* | 077 | 134 | 135 | 136 | 137 |

## Instruction Timing

Instruction timing is dependent upon whether a branch is taken. The decision to branch occurs at the second CP the new address is in the adder. Issue of a conditional branch is delayed if the accumulator is not free 3 CPs prior to the second CP that the instruction is in the adder.

If the branch is taken, instructions 100 through 137 have the same timing as the corresponding unconditional branch.

If the branch is not taken, the NIP register is valid the next CP after the second CP that the instruction is in the adder. Two CPs are then required for the instruction to advance to the CIP register and issue.

## Description

Instructions 100 through 137 jump to a new program sequence only if a branch condition is met. Instructions 070 through 077 represent the eight branch modes in the unconditional form. All possibilities of the eight modes are combined with four branch criteria to form the set of instructions 100 through 137.

The C = 0 branch condition causes the branch to be taken if the carry flag equals 0. If the carry flag equals 1, the current program sequence continues.

The C ≠ 0 branch condition causes the branch to be taken if the carry flag equals 1. If the carry flag equals 0, the current sequence continues.

The A = 0 branch condition causes the branch to be taken if the accumulator content equals 0. If the accumulator content does not equal 0, the current sequence continues.

The A ≠ 0 branch condition causes the branch to be taken if the accumulator content does not equal 0. If the accumulator content does equal 0, the current sequence continues.

## Instructions 140 through 147 and 154 through 157

Instructions 140 through 177 allow IOP control of the I/O channel activity. The contents of the *d* field of the instruction specifies the channel instructions 140 through 157. Refer to Table 7-6 for a list of instructions 140 through 147 and 154 through 157.

Table 7-6. Instructions 140 through 147 and 154 through 157

| Machine Instruction | ELAN | Description |
|---|---|---|
| 140 | iod : 0 | Channel *d* function 0 |
| 141 | iod : 1 | Channel *d* function 1 |
| 142 | iod : 2 | Channel *d* function 2 |
| 143 | iod : 3 | Channel *d* function 3 |
| 144 | iod : 4 | Channel *d* function 4 |
| 145 | iod : 5 | Channel *d* function 5 |
| 146 | iod : 6 | Channel *d* function 6 |
| 147 | iod : 7 | Channel *d* function 7 |
| 154 | iod : 14 | Channel *d* function 14 |
| 155 | iod : 15 | Channel *d* function 15 |
| 156 | iod : 16 | Channel *d* function 16 |
| 157 | iod : 17 | Channel *d* function 17 |

**Issue Conditions**

Accumulator free 1 CP after issue
No I/O function in progress

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field and function bits to channel decode |
| CP 1 | Channel decode and Go I/O (internal verses external)<br>Transmit accumulator bits $2^0$ through $2^7$ to function fanout |
| CP 2 + M | Fanout bits $2^0$ through $2^7$<br>Transmit accumulator bits $2^8$ through $2^{15}$ to function fanout |
| CP 3 + M | Fanout bits $2^8$ through $2^{15}$<br>Accumulator free |

### Instruction Timing (continued)

| | |
|---|---|
| CP 4 + M + N | -- |
| CP 5 + M + N | Function complete |
| CP 6 + M + N | Next I/O function issue |

For functions 14 and 15, a conflict may occur that can delay the accumulator free and/or issue of a subsequent I/O function by M. This delay can vary from 0 to 6 CPs.

For the remaining functions, a conflict may occur that can delay the issue of a subsequent I/O function by N. This delay can vary from 0 to 9 CPs.

### Description

Instructions 140 through 147 and 154 through 157 send the contents of the accumulator to the channel interface designated by the *d* field. The *d* field instructions 140 through 147 and 154 through 157 specify the I/O channel.

An I/O instruction sends the low-order 4 bits of the function code and a Go-function signal to the channel interface control circuitry. The channel interface interprets the 4-bit code in a manner unique to that channel. Refer to Section 5, "Channel Adapters," for information on the function-code format.

**Note:** I/O functions can have delays caused by path or resource conflicts depending on the channel number and function type. Non-status functions can issue every 6 CPs. Status functions for channels 10 through 37, functions 12 and 13 can issue every 15 CPs.

Status and non-status functions can be delayed by up to 9 CPs if a local memory data transfer conflict occurs. This occurs if a function is issued to a channel and that same channel is in the process of outputting a parcel of local memory data.

## Instructions 150 through 153

Instructions 150 through 153 are used to send data to the interface defined by the *d* field. Refer to Table 7-7 for a list of instructions 150 through 153.

Table 7-7. Instructions 150 through 153

| Machine Instruction | ELAN | Description |
|---|---|---|
| 150 | iod : 10 | Channel *d* function 10 |
| 151 | iod : 11 | Channel *d* function 11 |
| 152 | iod : 12 | Channel *d* function 12 |
| 153 | iod : 13 | Channel *d* function 13 |

**Issue Conditions** Accumulator free 1 CP after issue

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field and function bits to channel decode |
| CP 1 | Channel decode and Go I/O function (internal verses external)<br>Transmit accumulator bits $2^0$ through $2^7$ to function fanout |
| CP 2 | Fanout bits $2^0$ through $2^7$<br>Transmit accumulator bits $2^8$ through $2^{15}$ to function fanout |
| CP 3 | Fanout bits $2^8$ through $2^{15}$ |
| CP 4 | -- |
| CP 5 | -- |
| CP 6 | -- |
| CP 7 | -- |
| CP 8 | -- |
| CP 9 | -- |
| CP 10 | Enter accumulator contents with status channels 0 through 7, functions 10 through 13<br>Accumulator free |

## Instruction Timing (continued)

CP 11 + M Enter accumulator contents with status channels 10 through 37, functions 10 through 11
Accumulator free

CP 12 --

CP 13 --

CP 14 --

CP 15 + N Enter accumulator contents with status channels 10 through 37, functions 12 through 13
Accumulator free

For functions 10 and 11, channels 10 through 37, a conflict may occur that can delay the accumulator free by M. This delay can vary from 0 to 6 CPs.

For function 12 and 13, channels 10 through 37, a conflict may occur that can delay the accumulator free by N. This delay can vary from 0 to 9 CPs.

## Description

Instructions 150 through 153 transfer a 16-bit quantity from the channel interface into the accumulator. The value transferred depends on which channel interface is used. These instructions send data to the interface defined by the *d* field. The interface responds by returning a value to the accumulator. The carry flag clears.

An I/O instruction sends the low-order 4 bits of the function code and a Go-function signal to the channel interface control circuitry. The channel interface interprets the 4-bit code in a manner unique to that channel. Refer to Section 5, "Channel Adapters," for information on the function-code format.

**Note:** I/O functions can have delays caused by path or resource conflicts depending on the channel number and function type. Non-status functions can issue every 6 CPs. Status functions for channels 10 through 37, functions 12 and 13 can issue every 15 CPs.

Status and non-status functions can be delayed by up to 9 CPs if a local memory data transfer conflict occurs. This occurs if a function is issued to a channel and that same channel is in the process of outputting a parcel of local memory data.

## Instructions 160 through 167 and 174 through 177

Instructions 160 through 167 and 174 through 177 are used to send data to the interface defined by the channel number in the B register. Refer to Table 7-8 for a list of instructions 160 through 167 and 174 through 177.

Table 7-8. Instructions 160 through 167 and 174 through 177

| Machine Instruction | ELAN | Description |
|---|---|---|
| 160 | IOB:0 | Channel B function 0 |
| 161 | IOB:1 | Channel B function 1 |
| 162 | IOB:2 | Channel B function 2 |
| 163 | IOB:3 | Channel B function 3 |
| 164 | IOB:4 | Channel B function 4 |
| 165 | IOB:5 | Channel B function 5 |
| 166 | IOB:6 | Channel B function 6 |
| 167 | IOB:7 | Channel B function 7 |
| 174 | IOB:14 | Channel B function 14 |
| 175 | IOB:15 | Channel B function 15 |
| 176 | IOB:16 | Channel B function 16 |
| 177 | IOB:17 | Channel B function 17 |

**Issue Conditions**

Accumulator free in 1 CP after issue
B register not reserved
No I/O function in progress

**Instruction Timing**

| | |
|---|---|
| CP 0 | Issue<br>Transmit *d* field and function bits to channel decode |
| CP 1 | Channel decode and Go I/O (internal verses external). Transmit accumulator bits $2^0$ through $2^7$ to function fanout |
| CP 2 + M | Fanout bits $2^0$ through $2^7$<br>Transmit accumulator bits $2^8$ through $2^{15}$ to function fanout |

**Instruction Timing (continued)**

| | |
|---|---|
| CP 3 + M | Fanout bits $2^8$ through $2^{15}$<br>Accumulator free |
| CP 4 + M + N | -- |
| CP 5 + M + N | Function complete |
| CP 6 + M + N | Next I/O function issue |

For functions 14 and 15, a conflict may occur that can delay the accumulator free and/or issue of a subsequent I/O function by M. This delay can vary from 0 to 6 CPs.

For the remaining functions, a conflict may occur that can delay the issue of a subsequent I/O function by N. This delay can vary from 0 to 9 CPs.

**Description**

Instructions 160 through 167 and 174 through 177 send data to the interface defined by the channel number in the B register.

An I/O instruction sends the low-order 4 bits of the function code and a Go-function signal to the channel interface control circuitry. The channel interface interprets the 4-bit code in a manner unique to that channel. Refer to Section 5, "Channel Adapters," for information on the function-code format.

**Note:** I/O functions can have delays caused by path or resource conflicts depending on the channel number and function type. Non-status functions can issue every 6 CPs. Status functions for channels 10 through 37, functions 12 and 13 can issue every 15 CPs.

Status and non-status functions can be delayed by up to 9 CPs if a local memory data transfer conflict occurs. This occurs if a function is issued to a channel and that same channel is in the process of outputting a parcel of local memory data.

Instructions 170 through 173 are used to send data to the interface defined by the channel number in the B register. Refer to Table 7-9 for a list of instructions 170 through 173.

Table 7-9. Instructions 170 through 173

| Machine Instruction | ELAN | Description |
|---|---|---|
| 170 | IOB:10 | Channel B function 10 |
| 171 | IOB:11 | Channel B function 11 |
| 172 | IOB:12 | Channel B function 12 |
| 173 | IOB:13 | Channel B function 13 |

**Issue Conditions**

Accumulator free in 1 CP after issue
B register not reserved

**Instruction Timing**

CP 0 Issue
Transmit *d* field and function bits to channel decode

CP 1 Channel decode and Go I/O function (internal verses external)
Transmit accumulator bits $2^0$ through $2^7$ to function fanout

CP 2 Fanout bits $2^0$ through $2^7$
Transmit accumulator bits $2^8$ through $2^{15}$ to function fanout

CP 3 Fanout bits $2^8$ through $2^{15}$

CP 4 --

CP 5 --

CP 6 --

CP 7 --

CP 8 --

CP 9 --

CP 10 Enter accumulator contents with status channels 0 through 7, functions 10 through 13
Accumulator free

## Instruction Timing (continued)

CP 11 + M Enter accumulator contents with status channels 10 through 37, functions 10 through 11
Accumulator free

CP 12 --

CP 13 --

CP 14 --

CP 15 + N Enter accumulator contents with status channels 10 through 37, functions 12 through 13
Accumulator free

For functions 10 and 11, channels 10 through 37, a conflict may occur that can delay the accumulator free by M. This delay can vary from 0 to 6 CPs.

For functions 12 and 13, channels 10 through 37, a conflict may occur that can delay the accumulator free by N. This delay can vary from 0 to 9 CPs.

## Description

Instructions 170 through 173 transfer a 16-bit quantity into the accumulator. The value transferred depends on which channel interface is used. These instructions send data to the interface defined by the channel number in the B register. The interface responds by returning a value to the accumulator. The carry flag clears.

An I/O instruction sends the low-order 4 bits of the function code and a Go-function signal to the channel interface control circuitry. The channel interface interprets the 4-bit code in a manner unique to that channel. Refer to Section 5, "Channel Adapters," for information on the function-code format.

**Note:** I/O functions can have delays caused by path or resource conflicts depending on the channel number and function type. Non-status functions can issue every 6 CPs. Status functions for channels 10 through 37, functions 12 and 13 can issue every 15 CPs.

Status and non-status functions can be delayed by up to 9 CPs if a local memory data transfer conflict occurs. This occurs if a function is issued to a channel and that same channel is in the process of outputting a parcel of local memory data.

# BIBLIOGRAPHY

## Cray Research, Inc. Publications

Submit orders for Cray Research, Inc. publications to the following address:

Cray Research, Inc.
Logistics
6251 South Prairie View Road
Chippewa Falls, WI 54729

*IOS Model E Hardware Reference Booklet*, Cray Research publication number CQH-1011-000.

The *IOS Model E Hardware Reference Booklet* contains IOS-E instruction descriptions, channel functions, and channel function diagrams. This booklet is for Cray Research field engineers and analysts.

## Original Equipment Manufacturers' Publications

Contact your local IBM representative, or IBM branch office to order the following publication.

*IBM System/360 and System 370 I/O Interface Channel to Control Unit Original Equipment Manufacturers' Information*, International Business Machines (IBM) Corporation document number GA22-6974-09.

# Reader Comment Form

**Title:** **IOS Model E System Programmer Reference Manual**

**Number:** **CSM-1010-000**

Your feedback on this publication will help us provide better documentation in the future. Please take a moment to answer the few questions below.

For what purpose did you primarily use this manual?

_____Troubleshooting
_____Tutorial or introduction
_____Reference information
_____Classroom use
_____Other - please explain ____________________

Using a scale from 1 (poor) to 10 (excellent), please rate this manual on the following criteria and explain your ratings:

_____Accuracy ____________________
_____Organization ____________________
_____Readability ____________________
_____Physical qualities (binding, printing, page layout) ____________________
_____Amount of diagrams and photos ____________________
_____Quality of diagrams and photos ____________________

Completeness (Check one)

_____Too much information ____________________
_____Too little information ____________________
_____Just the right amount of information

Your comments help Hardware Publications and Training improve the quality and usefulness of your publications. Please use the space provided below to share your comments with us. When possible, please give specific page and paragraph references. We will respond to your comments in writing within 48 hours.

NAME____________________
JOB TITLE____________________
FIRM____________________
ADDRESS____________________
CITY__________STATE__________ZIP__________
DATE____________________

[or attach your business card]

CRAY
RESEARCH, INC.

CUT ALONG THIS LINE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**BUSINESS REPLY CARD**
FIRST CLASS PERMIT NO 6184 ST. PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY
RESEARCH, INC.

**Attn: Hardware Publications & Training**
**770 Industrial Boulevard**
**Chippewa Falls, WI 54729**

STAPLE