CRAY RESEARCH, INC.

CRAY X-MP™ AND CRAY-1® COMPUTER SYSTEMS

I/O SUBSYSTEM MODEL B HARDWARE REFERENCE MANUAL

HR-0030

RECORD OF REVISION

PUBLICATION NUMBER HR-0030

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.

Every page changed by a reprint with revision has the revision level in the lower righthand corner. Changes to part of a page are noted by a change bar in the margin directly opposite the change. A change bar in the margin opposite the page number indicates that the entire page is new. If the manual is rewritten, the revision level changes but the manual does not contain change bars.

Requests for copies of Cray Research, Inc. publications should be directed to the Distribution Center and comments about these publications should be directed to:

CRAY RESEARCH, INC.
2520 Pilot Knob Road
Suite 310
Mendota Heights, Minnesota 55120

Revision	Description
	October 1982 - Original printing; this publication and publication HR-0029, the CRAY-1 S Series Mainframe Reference Manual, obsolete publication HR-0808.
01	December 1983 - This change packet incorporates information pertaining to the use of the Memory (100 Mbyte) Channel with a Solid-state Storage Device and changes in how system errors are reported in machines with serial numbers of 21 and higher.
02	February 1985 - This change packet adds information for the DD-49 Disk Storage Unit, corrects information for instructions 047, 065, and 077, and corrects other minor technical errors.
A	October 1985 - Reprint with revision. Information from change packets 01 and 02 has been incorporated. No other changes were made.
B	May 1986 - Reprint with revision. Disk storage unit interface information has been removed, and is now in HR-0077, Disk Systems Hardware Reference Manual. In addition, miscellaneous technical and editorial changes have been made. All previous printings are obsolete.

PREFACE

This publication describes the operation of the Cray Research, Inc. I/O Subsystem (IOS), which handles external communication and mass storage for Cray Research computer systems. This publication is written to assist programmers and engineers, and assumes the reader has a familiarity with digital computers.

The four sections of an I/O Processor are described in this publication. These four sections are the memory section, control section, computation section, and input/output section. Also described in this publication are instructions and their formats, interface requirements to peripheral devices, and Buffer Memory of an IOS.

Additionally, appendixes contain detailed reference information.

Details of Cray Research, Inc., mainframes that use the Cray IOS are given in the following manuals:

HR-0029	CRAY-1 S Series Mainframe Reference Manual
HR-0032	CRAY X-MP Dual-processor Mainframe Reference Manual
HR-0064	CRAY-1 M Series Mainframe Reference Manual
HR-0088	CRAY X-MP Single-processor Mainframe Reference Manual
HR-0097	CRAY X-MP Four-processor Mainframe Reference Manual

Details of how the IOS interfaces to DD-29, DD-39, and DD-49 Disk Storage Units are provided in CRI publication HR-0077, Disk Systems Hardware Reference Manual.

//

WARNING

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

//

CONTENTS

FIGURES

TABLES

TABLES (continued)

INDEX

GENERAL INFORMATION 1

The Cray Research, Inc. Input/Output Subsystem (IOS) provides high-capacity data communications between Central Memory of a Cray mainframe and peripheral devices, storage devices, and front-end computers. The IOS is a required component of the CRAY-1 S series computer systems, models S/1200 through S/4400; all models of the CRAY X-MP series of computer systems; and all models of the CRAY-1 M series of computer systems. System overviews are provided in the relevant mainframe reference manuals.

The IOS is housed in a chassis (see figure 1-1) similar to a Cray Research, Inc. mainframe chassis and is composed of two to four I/O Processors, a group of interfaces, and Buffer Memory.

An I/O Processor (IOP) is a fast, multipurpose computer capable of transferring data at extremely high rates. A 16-bit processor and a fast bipolar Local Memory combine to support high-speed I/O operations by an IOP. (Local Memory, contained in each IOP, is distinguished from Buffer Memory, which is common to all IOPs.) The input and output capabilities make the IOP useful for network control, mass storage access, and computer interfacing.

An I/O Subsystem has a minimum of two IOPs. The Master I/O Processor (MIOP) and a Buffer I/O Processor (BIOP) are mandatory. In addition, the IOS can have one or two Disk I/O Processors (DIOPs), permitting a maximum of 48 disk units to be connected to the system. An Auxiliary I/O Processor (XIOP) controls block multiplexer channels and can be selected as an alternative to a DIOP.

An example of a four-processor IOS configuration is shown in figure 1-2. Configurations are factory hard-wired. Available configurations are described in the relevant mainframe reference manuals.

I/O PROCESSOR FUNCTIONS

In the I/O Subsystem, each IOP runs independently and is responsible for its own set of functions. IOP functions are defined by the way the IOP is attached to the other IOPs (and the mainframe) and by the peripheral equipment attached to it.

Software in each IOP performs specific functions and is structured to perform its tasks as efficiently as possible. Each IOP logs information and keeps statistics about channel use and error detection and recovery. The IOPs share Buffer Memory for communicating with each other. By using this communication mechanism, the processors can perform functions for each other.

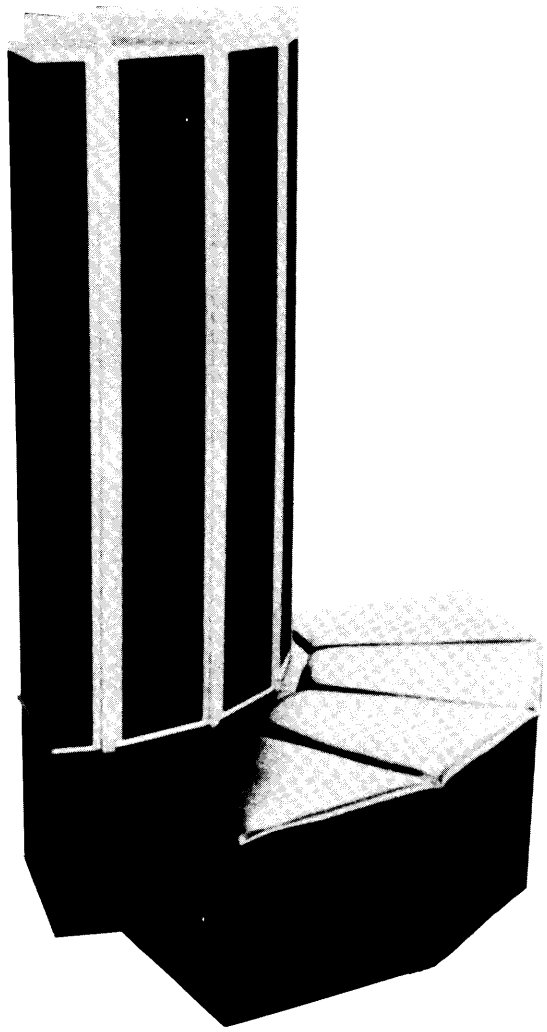
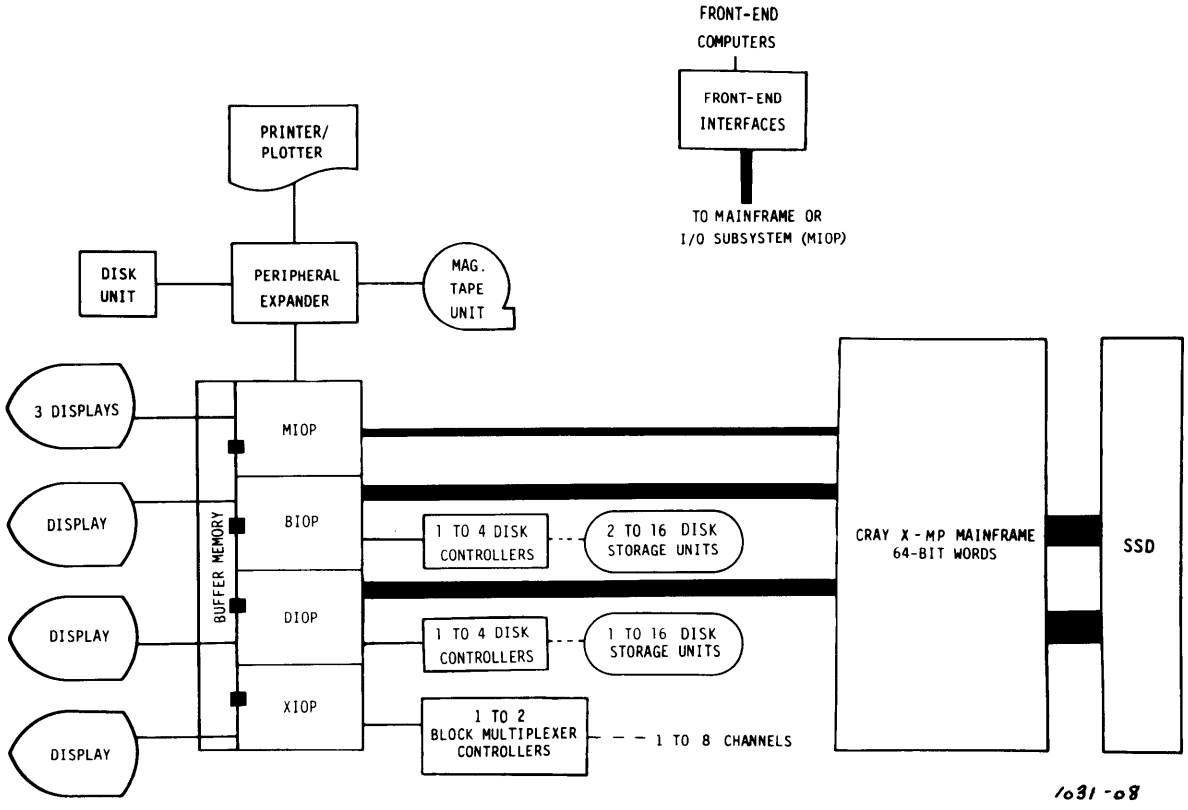


Figure 1-1. I/O Subsystem Chassis



6 Mbyte channel
100 Mbyte channel
1250 Mbyte channel

Figure 1-2. Four Processor IOS Configuration

MASTER I/O PROCESSOR RESPONSIBILITIES

The MIOP handles all communication with the Cray mainframe, including disk and tape requests and station communications. The MIOP supports front-end and station software, and handles input and output operations on the Peripheral Expander channel.

The MIOP is the first IOP to be deadstarted. The MIOP initializes the contents of Buffer Memory and deadstarts the other processors in the configuration. The MIOP, with the BIOP and the disk handling capabilities of the DIOP, deadstarts the Cray mainframe.

BUFFER I/O PROCESSOR RESPONSIBILITIES

The BIOP transfers data between Central Memory and Buffer Memory across a Memory (100 Mbyte) Channel. The BIOP also performs disk I/O to and from disk units attached to the BIOP's channels. I/O is driven by messages from the mainframe. (IOS software supports DD-29, DD-39 and DD-49 Disk Storage Units.) The BIOP performs error recovery when errors are detected on data transfers.

DISK I/O PROCESSOR RESPONSIBILITIES

The DIOP performs disk I/O to and from disk units attached to the DIOP's channels. I/O is driven by messages from the mainframe. The DIOP performs error recovery when errors are detected on data transfers. If equipped with a Memory (100 Mbyte) Channel, a DIOP transfers data between Central Memory and Buffer Memory.

AUXILIARY I/O PROCESSOR RESPONSIBILITIES

The XIOP's main function is to handle data from IBM-compatible tape drives and buffer the data to Buffer Memory. The XIOP can also be used to drive other peripherals. I/O is driven by messages from the mainframe. The XIOP performs error recovery procedures when errors are detected while transferring data to or from tape.

I/O PROCESSOR ORGANIZATION

Each IOP has a memory section, a control section, a computation section, and an input/output section (figure 1-3). A brief description of each section follows.

LOCAL MEMORY SECTION

IOP memory is called Local Memory. It consists of four sections of 4 banks each of random access, solid-state memory. All memory sections function independently of each other.

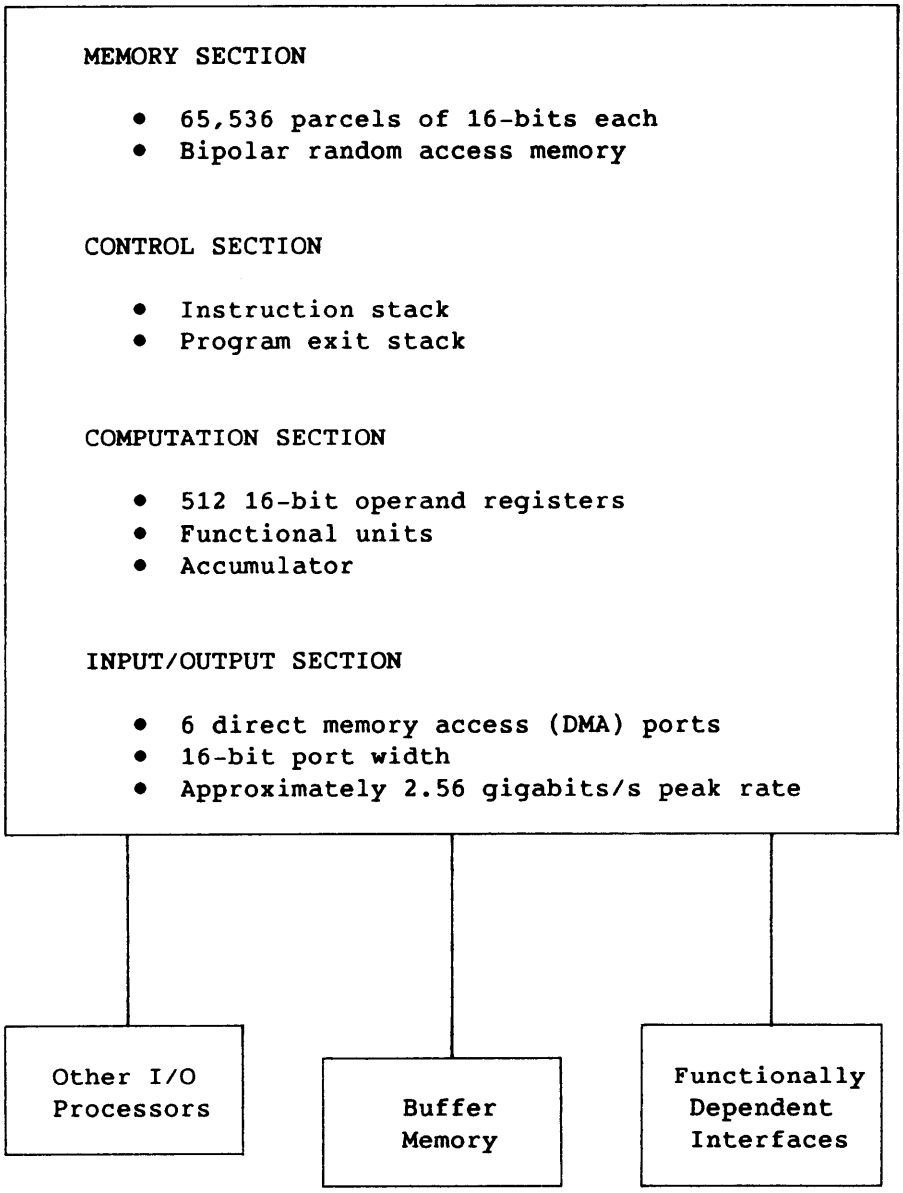


Figure 1-3. Basic Organization of an I/O Processor

Memory cycle time is 4 clock periods (CPs), which means a section is usable again 4 CPs after the preceding memory reference. An exception is the I/O write operation which requires 6 CPs. Access time (the time required to bring an operand from memory to the accumulator) is 7 CPs. Memory capacity is fixed at 65,536 units (called parcels) of 16 bits each plus 2 odd parity bits for each parcel.

See section 2 for additional information on Local Memory.

CONTROL SECTION

An IOP control section has an instruction stack, program exit stack, and control logic. An IOP has a 12.5 ns clock period and executes 128 instruction codes as 16-bit (1-parcel) or 32-bit (2-parcel) instructions. Branching and I/O instructions are included in the set. Instructions are stored in memory, and transferred into the instruction stack under the control of a program address counter. Instructions issue from the instruction stack and are decoded into the control signals enabling the functions of the instruction.

The instruction stack is a 32-parcel circular buffer that provides fast access to a moving window of program instructions. The longest held parcels are overwritten by newly transferred instructions. The program is free to branch quickly about inside the stack, only slowing when more instructions must be read from memory.

The program exit stack is a last-in-first-out set of 16 registers that stores return addresses for program subroutine calls. The 16 registers provide for 14 nested levels of subroutines in the program. A pointer keeps track of the levels involved. An interrupt is generated when the stack is emptied or filled. Since the stack may be loaded or unloaded by the program, an unlimited number of subroutines can be nested by means of the software.

See section 3 for additional information on IOP control.

COMPUTATION SECTION

The IOP computation section contains operand registers, functional units, and an accumulator operating together to execute a program of instructions stored in memory.

All arithmetic (addition and subtraction) is in twos complement mode in one Adder functional unit. Floating-point arithmetic is not incorporated. A Shifter functional unit provides left or right shifts of up to 31 bit positions, either circular or end-off shifting. A logical product operation is provided.

Any of 512 operand registers are used for temporary data storage or for indirect memory addressing. All operand registers are 16 bits wide.

The accumulator is a 16-bit signed register that temporarily stores operands or results. All data movement within the single-address IOP uses the accumulator either as a source of data or as the destination for results. An instruction code specifies the memory address or register from which the accumulator accesses data or to which the accumulator sends data.

The accumulator is also used for all transfers between memory and operand registers. Accumulator data can be sent to or received from specialized I/O channels. An accumulator bit occupies the 2^{16} bit position as the carry flag. When operations in the Adder yield a carry, the accumulator 2^{16} bit is toggled. The accumulator bit is included in all shift operations. The carry flag is available for conditional testing and can be set to reflect I/O channel status.

See section 4 for additional information on the computation section of an I/O Processor.

INPUT/OUTPUT SECTION

Communication with an IOP is through six direct memory access (DMA) ports. Each port is bidirectional and can transfer four 16-bit parcels every 6 CPs. An I/O Processor has a maximum of 40 channels. Input and output channels can be active at the same time, as long as they use different ports and reference different memory sections. The ports are assigned to channels with the possibility of several channels sharing one port. The slower the required data rate on the channels, the more channels can be multiplexed into one DMA port.

Channels use Busy and Done flags to signal the IOP and communicate directly with the IOP accumulator for control information. Use of the flags and the accumulator varies with the specific design of the channel interface logic. All channels communicate status and functions through the accumulator. Some low-speed devices can transfer data directly to and from the accumulator using one of the channel registers.

Section 5 describes the input/output section of an I/O Processor.

REQUIRED INTERFACES

Interfaces adapt an IOP to peripheral devices. Section 7 describes the required interfaces for a Cray I/O Subsystem.

BUFFER MEMORY

Buffer Memory assists data transfer between peripheral devices and Central Memory in a Cray mainframe. Buffer Memory is housed in the IOS chassis and stores 1 million, 4 million, or 8 million 64-bit words. All I/O Processors share Buffer Memory, which uses single-error correction/double-error detection (SECDED) data protection.

Section 8 provides additional information on Buffer Memory.

I/O SUBSYSTEM CLOCK

The clock controlling the I/O Subsystem is a crystal-controlled oscillator that runs at 80 MHz and gives a clock period (CP) of 12.5 ns. The speed can be adjusted slightly higher or lower for maintenance purposes. When operations require exact timing information, such as interval timing with the real-time clock, maintenance personnel should be contacted to verify the clock period.

CONVENTIONS

In this manual, parentheses are used in instructions as a form of shorthand notation for the expression "the contents of operand register ---" or "the contents of memory location ---." For example, A = A - (B) means "Subtract the contents of the operand register addressed by the B register from the previous accumulator content." Special symbols used in instruction descriptions are given in section 6.

Register bit positions are numbered from right to left as powers of 2, starting with bit 2^0.

Channel numbers are specified in octal. All other numbers, unless otherwise indicated, are decimal numbers. Octal numbers other than channel numbers are indicated with a subscript 8 (for example, 12_8).

LOCAL MEMORY SECTION 2

I/O Processor (IOP) memory, called Local Memory, consists of 16 banks of random access, solid-state storage. The 16 banks of 4096 parcels each are divided into four sections of 4 banks each. Each 4-bank section is independent of the other sections. Local Memory has the following characteristics:

- 65,536 parcels of 16 bits
- 16 banks of 4,096 parcels each
- 4 clock-period (CP) bank cycle time on an I/O read operation or accumulator reference
- 6 CP bank cycle time on an I/O write operation
- Access time every 6 CPs
- Read operation to accumulator in 7 CPs
- One instruction fetched per CP
- 1 data parcel per CP on sequential addressing for fetch or I/O
- Dual odd parity protection
- Six full-duplex, direct memory access (DMA) ports

Local Memory speeds, organization, access, addressing, and data protection are described in this section.

LOCAL MEMORY SPEEDS

For an accumulator read or write reference, Local Memory has a cycle time of 4 CPs. For an I/O reference, Local Memory cycle time is 4 CPs on a read operation and 6 CPs on a write operation. Access time, the time required to fetch an operand from Local Memory to the accumulator, is 7 CPs. Instructions are fetched from Local Memory at the rate of 1 parcel per CP in 4-parcel bursts. I/O operations transfer data in bursts of 4 parcels, 1 parcel each CP. Local Memory organization (explained later in this section) allows sequential addresses to be referenced every CP, whether the operation is a read or a write. However, the same section can only be referenced once every 6 CPs by the same DMA port. Data transferred in the operation can be 1 parcel of operand data or 4 parcels of I/O data. The Peripheral Expander is an exception; it transfers 1 parcel of I/O data per reference.

LOCAL MEMORY ORGANIZATION

Each 4-bank section of Local Memory has separate access paths, sequence controls, write data registers, and read data registers. The 4 banks within a section share a common sequence control. Reference to any one bank of the section also references the other three banks in the section, but only the addressed bank transfers data.

A read reference to a section in Local Memory causes all 4 banks in that section to read out parcels to their read registers. The addressed parcel or parcels are then gated to their destination. In an operand reference, a single parcel goes to the accumulator. In an I/O reference, each of the 4 parcels is gated in sequence to the output channel. The read reference and transmission sequence are fixed with reference to timing of the initiation and moving of data. Once the reference has been started, it continues automatically. After 4 CPs, the section is free to begin another reference.

A write sequence from an IOP's input/output section can be assembling data for a section of Local Memory in the bank write data registers at the same time an I/O read reference is reading data from that same section of Local Memory. If an I/O write follows any other write to the same section, a write register conflict occurs. The first write must complete before data can be sent for the subsequent write. This causes the second I/O write to take 6 CP, but only for this conflict case. The actual data moved per reference can vary from 1 operand parcel to a 4-parcel burst for an I/O operation.

LOCAL MEMORY ACCESS

Each Local Memory section has three 16-bit data paths for reading and two 16-bit data paths for writing. One read path and one write path go to the accumulator, which is a 16-bit signed register for temporary storage of operands and results. One read path and one write path go to the IOP's input/output section as direct memory access (DMA) ports. The last read path goes to the instruction stack and carries instruction parcels. The DMA ports are explained in greater detail in section 5, IOP input/output; the instruction stack is described in section 3, IOP control.

LOCAL MEMORY ADDRESSING

A parcel in Local Memory is addressed in 16 bits as shown in figure 2-1. The low-order 4 bits specify the section and bank of the section to be addressed. The high-order 12 bits select the storage chip and specify an address within the storage chip.

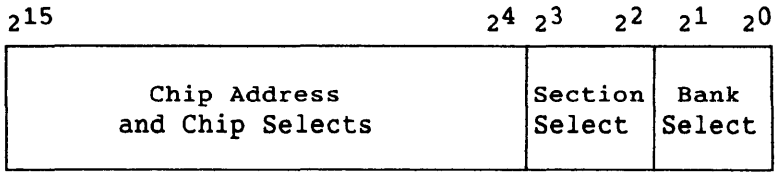


Figure 2-1. Local Memory Address Format

Three address paths go from each of the four Local Memory sections: one from the I/O section, one from the computation section, and one from the control section for instruction fetch references.

LOCAL MEMORY DATA PROTECTION

Data stored in Local Memory is protected by using a mechanism called voting memory. Each IOP has three separate copies of memory, and all three copies are referenced on each read and write. On reads, the copies are compared on a bit-by-bit basis. If the 3 bits do not agree, majority determines the value of the bit.

Each data word also has a parity bit. If a parity error exists after the majority vote operation, an error handling routine uses the Local Memory error channel to determine the location of the failing hardware and to issue this information for maintenance people. Normally, no further IOP operation is attempted until the Local Memory fault is corrected. The Local Memory error channel is described in section 5 of this publication.

IOP CONTROL SECTION 3

The control section of an I/O Processor (IOP) consists of an instruction stack and a program exit stack. The control section has the following characteristics:

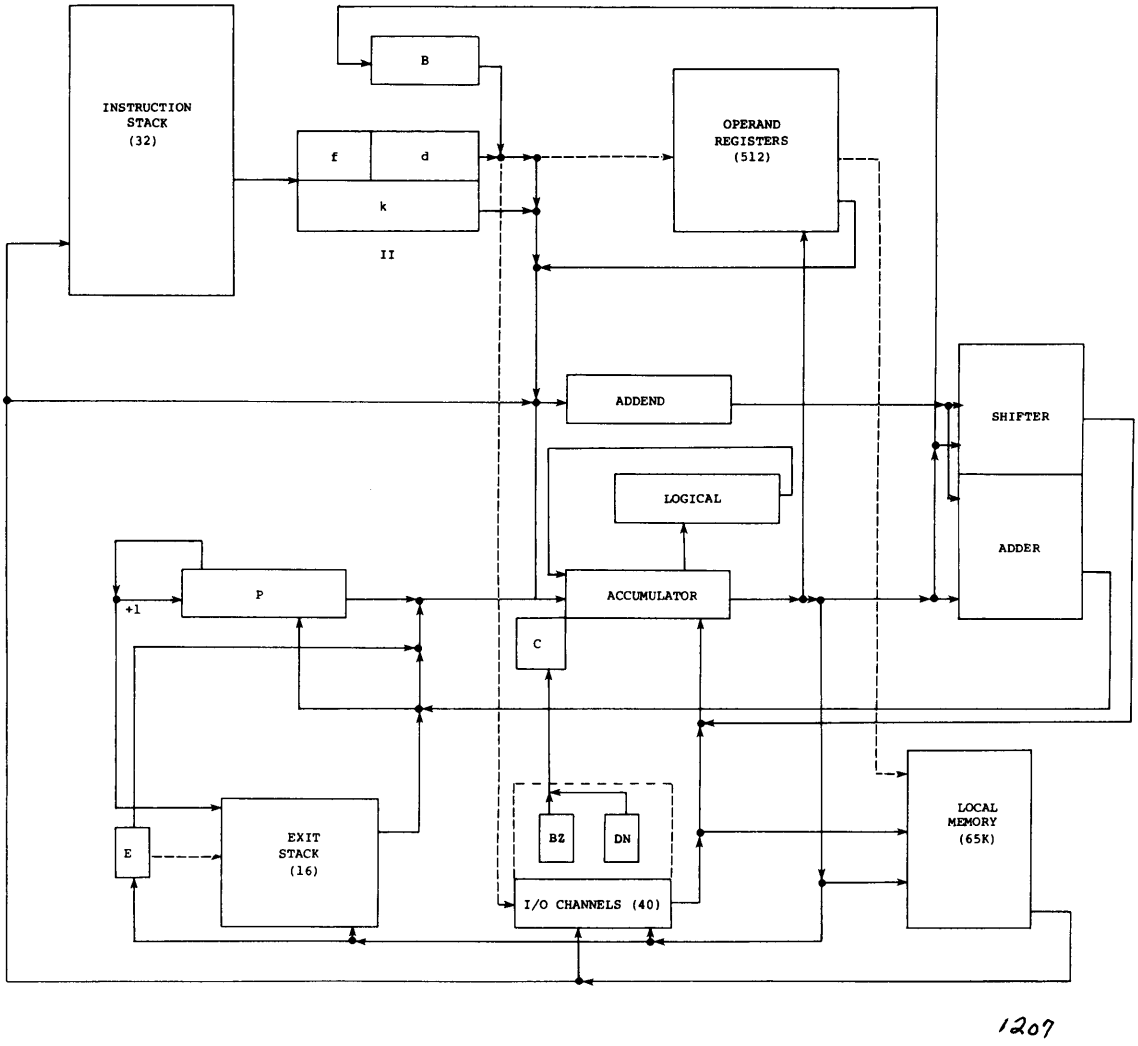
- 12.5 ns clock period (CP)
- Single addressing mode
- 128 operation codes
- Instruction stack
- Program exit stack, 16 primary levels

Instructions move data from a source to the accumulator and from the accumulator to a destination. Operand registers temporarily store operands and results. Functional units in the computation section of an IOP receive operand pairs and produce single results. One operand address is designated by the instruction, and the other operand is contained in the accumulator. Typically, data flows from Local Memory to the accumulator, from the accumulator (with an operand) to a functional unit, back to the accumulator, and from the accumulator to Local Memory.

Figure 3-1 is a block diagram of an IOP. Important features of the control section for the IOP are the instruction stack, II register, B register, P register, and exit stack. These features are described later in this section.

INSTRUCTION FORMATS

Instructions are 1 parcel (16 bits) or 2 parcels (32 bits) in size. A 1-parcel instruction consists of a function code (*f*) field and a designator (*d*) field. A 2-parcel instruction consists of the *f* and *d* fields and a constant (*k*) field. Figure 3-2 illustrates the formats of 1-parcel and 2-parcel instructions.

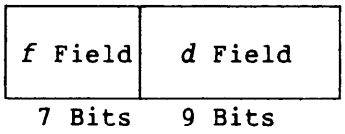


- - - Selection path

_____ Data path

Figure 3-1. I/O Processor Block Diagram

1-parcel instruction:



2-parcel instruction:

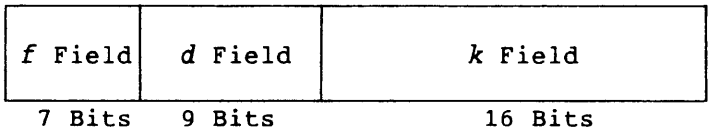


Figure 3-2. Instruction Formats

The 7-bit *f* field contains the instruction function code specifying the instruction to be executed and designating where and how the execution occurs. The 9-bit *d* field can contain data, address, or shift count and designates what machine resources the function of the *f* field is to use. The 16-bit *k* field is a constant field occupying the program parcel immediately following the *d* and *f* field parcel. A detailed explanation of the instruction formats is given in section 6.

INSTRUCTION STACK

Program instructions are fetched from Local Memory and stored in a 32-position, 16-bit instruction stack. The stack capacity for 32 instruction parcels allows short program loops to execute in the stack without reference to Local Memory. The instruction stack is 2 banks of registers with 16 parcels of program code in each bank (figure 3-3). Addresses alternate between the two banks so that loading of data from storage can interleave with the readout of instructions for execution. Instructions are fetched from Local Memory in bursts of 4 parcels; each burst references a storage address that is a multiple of 4. Loading is sequential once issue control selects which of the 4 parcels fetched is received first. If no memory conflicts exist, 1 parcel each clock period issues from Local Memory.

An instruction passes directly into execution if the instruction sequence can issue in that CP. The instruction is stored in the instruction stack whether it issues immediately or waits for issue. When an issue delay occurs, arriving parcels are also stored in the instruction stack.

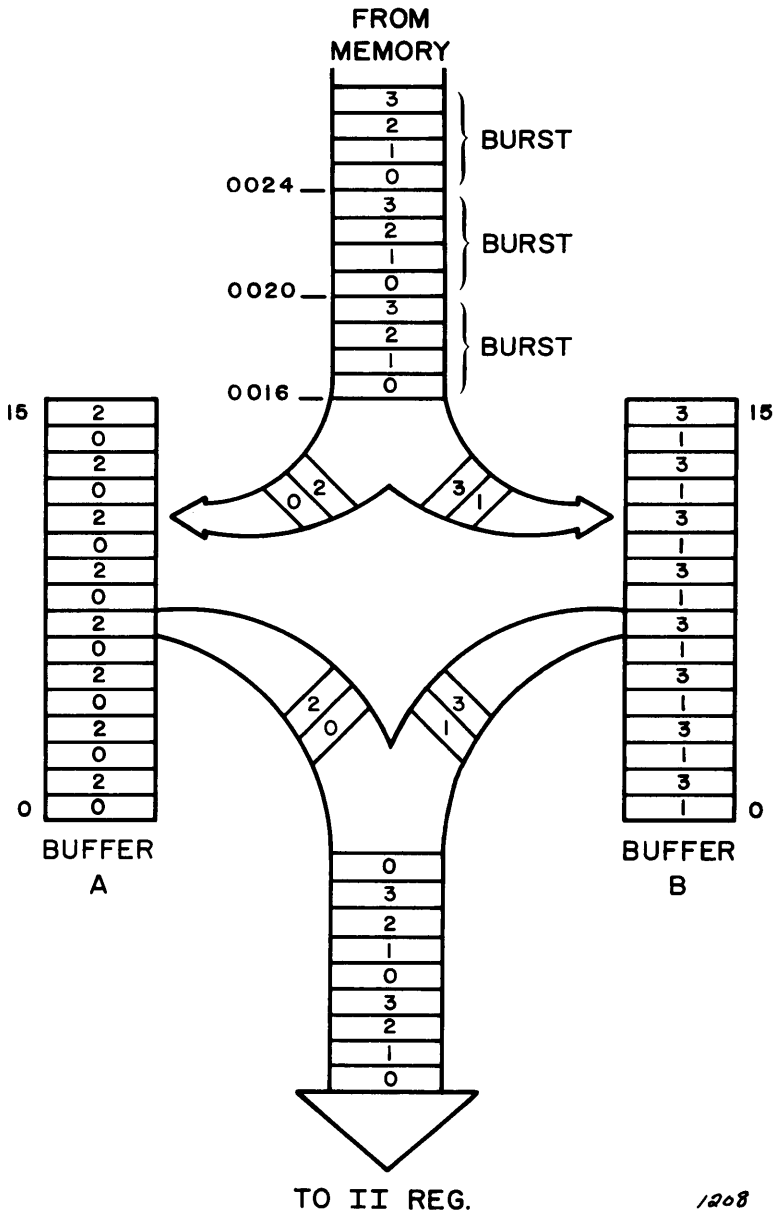


Figure 3-3. Instruction Stack Operation

The instruction stack is implemented as a circular buffer. When the stack is full and new program code is required, the new code is stored in the beginning address of the stack. Circular loading continues with the issue control circuits keeping a record of the current section of Local Memory represented in the instruction stack. To maintain a situation in which the next instructions to be executed are in the stack, internal (background) fetches are performed as instructions sequentially execute.

Out-of-stack branch conditions also cause a fetch, beginning at stack location 0. Once this fetch is accomplished, barring other branches, the internal fetch mechanism takes over. All absolute branches (instructions 074 through 077 and 120 through 137) are considered to be out of stack. Relative jumps (instructions 070 through 073 and 100 through 117) can be in-stack or out-of-stack, depending on the offset, the *d* field, and the locations of the stack and load pointers.

FORWARD RELATIVE BRANCH

Forward relative branches in the instruction stack that are less than or equal to an offset of 11_8 do not generate an out-of-stack condition. Forward branches with an offset greater than 11_8 may exceed the point that predetermined internal fetches have reached or will reach.

BACKWARD RELATIVE BRANCH

Backward relative branches in the instruction stack that are less than or equal to an offset of 13_8 do not generate an out-of-stack condition. Backward branches with an offset greater than 13_8 may exceed the point that predetermined internal fetches have reached or will reach. If the stack is being filled for the first time after an out-of-stack condition, a backward relative branch is only valid to location 0 of the stack.

INSTRUCTION ISSUE REGISTER

The Instruction Issue (II) register is a 16-bit register that receives the instruction parcel from the instruction stack. The instruction parcel can stay in the II register for more than 1 CP and leaves the II register when a new instruction is needed.

The operation code (*f* field) of the instruction parcel (bits 2^9 through 2^{15}) is translated by logic associated with the II register to determine the particular sequence of operations required. The *d* field of the instruction parcel (bits 2^0 through 2^8) is sent to the Register Pointer (RP) register, Destination Pointer (DP) register, Addend register, or accumulator. The *d* field also addresses an I/O channel and selects operand registers. If the *f* field translation shows the parcel to be the first of a 2-parcel instruction, the second parcel is sent from the instruction stack to the Addend register or accumulator and is not interpreted as an instruction.

B REGISTER

The B register is a 9-bit address register used to designate one of the 512 operand registers. The B register is loaded from the accumulator, taking the low-order (2^0 through 2^8) bits. Accumulator bit 2^0 goes into B register 2^0 location. The B register may also address the I/O channel for an I/O instruction or may be used as an operand. For I/O instructions, the B register contains the alternate low-order bits (bits 2^0 through 2^8) of the instruction *d* field. When used in this manner, the B register can be altered by the program, in contrast to the *d* field which is part of the program.

REGISTER POINTER REGISTER

The 9-bit Register Pointer (RP) register directly addresses one of the 512 operand registers for reading or writing. The RP register receives the *d* field from the II register or bits 2^0 through 2^8 of the B register on issue of each instruction using operand registers. The RP register is not shown in figure 3-1.

The operand registers are built to automatically read out data as addressed by the RP register, unless an instruction specifically demands a write into an operand register. The automatic read occurs each clock period and is ignored if the read data is not needed.

DESTINATION POINTER REGISTER

The 9-bit Destination Pointer (DP) register selects one of the 512 operand registers to receive the contents of the accumulator. It is accessed by an instruction type using the same operand register for both operand and result. The DP register receives the pointer from the *d* field or bits 2^0 through 2^8 of the B register when the instruction issues. A write operation from the accumulator to an operand register usually involves a delay between the time the instruction issues and the time the register pointer is required. The DP register stores the pointer during the delay period. The DP register is not shown in figure 3-1.

Because the transfer of a pointer from the DP register to the RP register uses the same path into the RP register as the pointer coming from the II register, instruction issue can be blocked until the path into the RP register is free.

PROGRAM ADDRESS REGISTER

The 16-bit Program Address (P) register holds the memory address of the instruction currently awaiting issue. The P register contents are automatically incremented as each instruction is executed in program sequence. A delay between reading from the instruction stack and instruction issue keeps the address in the P register two program steps behind the instruction stack readout address. This delay is transparent to the programmer.

Branch instructions alter the P register contents by either adding a positive or negative displacement value or by entering a new value.

PROGRAM EXIT STACK

The program exit stack stores the following:

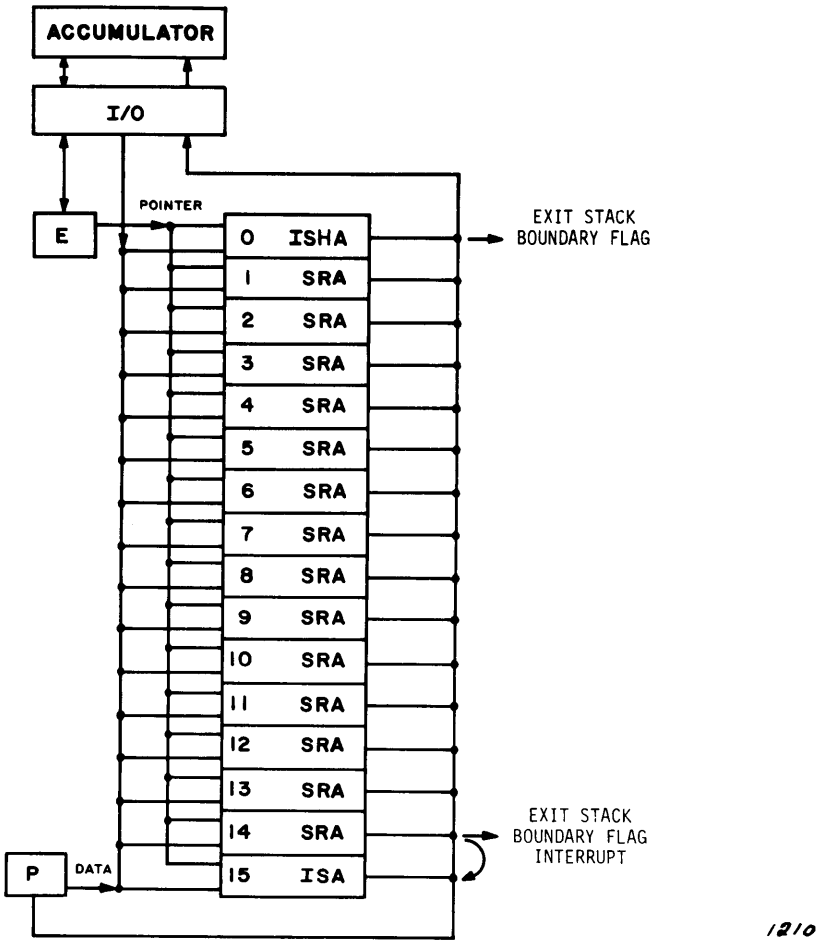
- A program return address when a subroutine is called
- A program return address when the program is suspended to handle interrupts
- The interrupt handler starting address

The program exit stack (figure 3-4) consists of sixteen 16-bit registers and is addressed by the E register, a 4-bit pointer. A program can access and modify the contents of the program exit stack and the E register through I/O channel functions.

Position 0 in the stack is reserved for the starting address of the interrupt handler. This address is entered in the stack by the deadstart program and remains unaltered during system execution. Interrupts cause the hardware to reference stack position 0 without reference to the E register.

SUBROUTINE CALLS

Positions 1 through 14 in the program exit stack are used for subroutine return addresses, referred to as SRA in figure 3-4. A subroutine call from a routine or an interrupt advances the E pointer by 1 and stores the routine's return address in the stack. On exit from the subroutine, the return address is read from the current location in the stack. Then the E pointer decrements by 1 to point to the next highest stack location. As a program goes into deeper levels of subroutines, the E count increases and more subroutine return addresses are stored. As the program exits from subroutines, the E count drops back toward 0.



ISHA Interrupt handler start address
SRA Subroutine return address or interrupted subroutine address
ISA Interrupted subroutine address

Figure 3-4. Program Exit Stack

When the E pointer approaches the full limit of the stack, an exit stack interrupt is generated to allow the software to reconfigure the stack. The sequence is as follows:

1. E reaches 13 following the execution of a return jump.

2. The new subroutine begins execution.

3. The next subroutine is called.

4. E goes to 14 due to a subroutine call.

5. New return address loads to stack position 14.

6. The program exit stack boundary flag is set, causing an interrupt.

7. Return jump to new subroutine enters a new value into the P register but does not jump.

8. Interrupt blocks instruction issue (if system and channel interrupts are enabled).

9. Stack loads the interrupted address (P) to position 15.

10. Interrupt handler begins executing.

When the E pointer reaches 0 and an exit instruction issues, an interrupt sets and the program jumps to the interrupt handler routine. The sequence is as follows:

1. E reaches 0.

2. Exit instruction occurs.

3. Interrupt sets.

4. Interrupt blocks further instruction issue (if system and channel interrupts are enabled).

5. Interrupt handler begins executing.

If return jumps are used in an interrupt handler, care must be taken that enough levels are available in the stack. An interrupt with the exit stack pointer at 13 moves the pointer to 14, leaving only one location open. A return jump causing a Program Fetch Request (PFR) interrupt may be issued with the stack pointer at 13. The return address goes into 14 and the interrupt address into 15, leaving two interrupts present--both the exit stack boundary and PFR, with the PFR being the higher priority and no stack locations available. If the E pointer is at 15 and an interrupt or subroutine call is generated, the pointer wraps around and returns to 0.

The program exit stack can be reconfigured by keeping the stack half full, allowing calls to deeper levels or exits to higher levels. The deepest level interrupt is characterized by E equal to 15. For this level of interrupt, the interrupt handler routine saves the higher half of the stack in memory (positions 1 through 7) and moves the lower half of the stack (positions 8 through 15) to the higher part of the stack (positions 1 through 8). Old position 15 holds the interrupted subroutine return address. When the reconfiguration is complete, the interrupt handler routine exits to the interrupted subroutine, and operation continues.

A reconfigured stack at the E equals 0 level exits directly to the interrupt handler routine. This highest level interrupt is characterized by E equal to 0 with the exit stack interrupt present. The interrupt handler routine can then rebuild the original higher half of the stack (reading it from memory), set E to 7, and exit to that subroutine return address.

NOTE

If E equals 13 and an interrupt occurs, E goes to position 14, but the Exit Stack Boundary flag does not set.

PROGRAM EXIT STACK AND I/O INTERRUPTS

An I/O interrupt is treated much like a subroutine call. The interrupted program address is stored in the program exit stack at the next stack position and the entrance address for the interrupt routine is read from position 0 of the stack. On servicing the interrupt, the hardware clears the System Interrupt Enable flag to prevent other interrupts from interrupting the handling routine.

If return jumps are used in the interrupt handling routine, enough locations must be left in the exit stack to handle the maximum number of levels encountered in the interrupt handling routine. When I/O is finished, the interrupt handler routine sets the System Interrupt Enable flag. The exit at the end of the interrupt handling routine reads the return address for the interrupted program from the exit stack and returns control to that point.

****

CAUTION

After any modification to the stack locations or the E pointer, at least 5 CPs must elapse before performing a program exit or a return jump or before enabling system interrupts. The same delay is necessary before using data read from the exit stack or E pointer. To achieve the required delay, perform three circular shifts of 17 positions on the accumulator before exiting the routine. Modifying stack locations or the E pointer should only be done when system interrupts are disabled.

****

PROGRAM FETCH REQUEST FLAG

The Program Fetch Request (PFR) flag is set during execution of jump instructions 074 through 077 and 120 through 137 when the instruction sequence finds a zero value in operand register *d*. This condition requests the monitor program to perform a function for this routine.

Setting the PFR flag suspends execution of the current program sequence with an interrupt request at the completion of the interrupted instruction. The monitor program then reads the value of the Channel 1 Interface Input register (which contains *d*) and interprets that number as identifying a particular request.

IOP COMPUTATION SECTION 4

An I/O Processor (IOP) adds, subtracts, left shifts, and right shifts by using the Adder and Shifter functional units. Temporary data storage is provided by a block of operand registers. All transfers to operand registers and all results from the functional units pass through the accumulator. The IOP computation section has the following characteristics:

- 16-bit architecture
- Twos complement arithmetic
- Integer addition/subtraction unit
- Shift unit
- Logical product
- 512 operand registers, each 16 bits wide

This section describes the computation section of an IOP. Refer to the I/O Processor block diagram (figure 3-1) showing the organization of the computation section of the IOP.

OPERAND REGISTERS

Computation in the IOP is supported by 512 operand registers. The operand registers act as temporary locations for data, as index registers, and as indirect memory address registers. Each operand register contains 16 bits of data and has a 1 clock period (CP) access time. The registers are addressed by the Register Pointer (RP) register (see section 3).

The only data path into operand registers is from the accumulator. Data leaving an operand register goes either to the accumulator or the Addend register as operand data, or it can go to the Memory Address (MA) register as memory address data.

NOTE

The operand registers are the only mechanism through which the computation section can reference memory.

MEMORY ADDRESS REGISTER

The 16-bit Memory Address (MA) register holds the address for a Local Memory reference. The MA register receives address information from an operand register and holds it for Local Memory use. This register is used for read and write memory references.

FUNCTIONAL UNITS

The IOP computation section has an Adder functional unit and a Shifter functional unit that perform all the arithmetic required by the instruction set.

ADDER FUNCTIONAL UNIT

IOP arithmetic is performed in the twos complement Adder functional unit. Adder operands come from the accumulator, instruction fields, B register, Program Address (P) register, operand registers, and Local Memory. Except for accumulator contents, operands come to the Adder through the Addend register. Adder results go to the accumulator for distribution as needed and to the P register. The 17-bit operands are received from the accumulator and returned to the accumulator; the seventeenth bit corresponds to the carry bit of the accumulator. Operands from the Addend register have 16 bits. Branch instructions use the Adder for P address calculations.

The Adder is also used for subtraction. In twos complement arithmetic, subtraction takes place by adding the ones complement of the subtrahend (the number subtracted from the minuend) to the minuend and then adding 1. When subtracting, the contents of the Addend register are inverted and passed to the Adder. The subtraction control signal is a 1, which is added to the intermediate result to give the final difference.

Both add and subtract computations take 1 CP; another CP is required to put the results into the accumulator and the Carry Bit register (described later in this section).

NOTE

For unsigned 16-bit arithmetic, the carry bit is toggled on $x - y$ if $x > y$ or $x = y$.

SHIFTER FUNCTIONAL UNIT

The Shifter functional unit implements IOP shifting instructions of up to 31 places left either circularly or end-off with zero fill. The Shifter receives the 17-bit accumulator data (including carry bit) to be shifted and the 5-bit (2^0 through 2^4) Addend register shift count. The Shifter accomplishes right shifting by using the inverted shift count. The shifted results (17 bits) are returned to the accumulator and the Carry Bit register (described later in this section). The 1 CP required for the shift is independent of the shift count and type of shift. Shift instructions require 2 CPs.

The maximum number of places an operand can be shifted is 31. If the shift count is 0, no shift occurs. If the count is greater than 16 for an end-off shift, the zero-filling clears the result. In all shifts, the carry bit is treated as the high-order bit (2^{16}) of the operand and the result.

ACCUMULATOR

The 16-bit accumulator register temporarily stores operands and results. Data from several sources can be routed to the accumulator; many destinations for accumulator contents are available. Sources and destinations are listed below.

Sources	Destinations
B register	B register
Operand registers	Operand registers
Adder/Shifter	Adder/Shifter
Memory	Memory
I/O channels	I/O channels
II register *d* field	
II register *k* field	

Logic at the input of the accumulator is enabled by the logical product instructions and creates the logical product of two operands: the accumulator and the input operand. The result goes directly to the accumulator with no extra time taken for the logical product function.

Program branch instructions require arithmetic to form the destination address from two operands. These instructions (070 through 137) do not alter the contents of the accumulator. Execution of these sequences is performed with a separate background accumulator not visible to the programmer.

CARRY BIT REGISTER

The 1-bit Carry Bit register holds the carry generated in the Adder or Shifter functional units. The carry bit is treated as if it were bit 2^{16} of the accumulator operand and is included in all add, subtract, and shift operations.

The carry bit is set by several conditional instructions that test I/O channel flags. The carry bit is also used as a criterion for many conditional jump and return jump instructions.

ADDEND REGISTER

The 16-bit Addend register supplies operands to the Adder and Shifter functional units. Whenever two operands are required, the accumulator supplies one operand and the Addend register supplies the other. The Addend register receives data from the B register, the instruction stack, the operand registers, or Local Memory. The Addend register sends data to the Adder and the Shifter functional units.

INPUT/OUTPUT SECTION 5

An I/O Processor (IOP) supports up to 40 channels for input or output (I/O) use. These 40 channels use six direct memory access (DMA) ports to Local Memory. Twelve channels and one port are assigned standard functions for the system, but the remainder are free for peripheral device or Cray mainframe support.

An IOP input/output section has the following characteristics:

- Supported by six full-duplex, direct memory access (DMA) ports
- Approximately 100 Mbytes/s per DMA port (maximum speed)
- 16 data bits, 2 status bits (Busy and Done)
- Channel number selected by instruction or B register contents
- Simultaneous input and output through separate ports

This section describes I/O configuration, I/O speeds, channel characteristics, standard channels, and the interrupt sequence.

I/O CONFIGURATION

The I/O channels are numbered in octal. I/O channels 0 through 13_8 are standard for all IOPs. Channels 14_8 through 47_8 are variable. For control purposes, functions can be sent to and statuses can be received from each IOP channel.

The interfaces associated with channels 14_8 through 47_8 have five DMA ports to Local Memory. (The sixth DMA port connects to Buffer Memory.) Faster devices connect to an IOP through the DMA ports, allowing block transfers. Slower devices can be supported by accumulator channels. Several devices can be interfaced to share a single DMA port, with each device assigned a unique channel number. This method supports groups of four disk storage units and Block Multiplexer Channels.

For interrupts, channels are assigned priorities that cause them to be serviced in the following order: Channel 0, Channel 1, and Channels 2 through 47_8 in descending priority.

I/O SPEEDS

Each DMA port can transfer a block of data at the approximate rate of 100 Mbytes/s. DMA ports can transfer data into Local Memory while other DMA ports simultaneously transfer data from Local Memory. One DMA port can transfer data into Local Memory and one DMA port can transfer data from Local Memory at a sustained rate of 100 Mbytes/s, barring Local Memory conflicts. If more than one port is transferring data either into or from Local Memory, speed degradation occurs.

The maximum speed of accumulator channels depends on the speed of the interrupt service routine.

CHANNEL CHARACTERISTICS

Operating characteristics for the accumulator channels and the channels using DMA ports are similar in many respects. The following descriptions outline the control and data signals used and give the requirements for each signal. The channels are described independently of the interfaces that can be connected to them. (Channel interfaces are described in section 7.)

ACCUMULATOR CHANNELS

The accumulator channels permit communication among the IOPs. Each accumulator channel uses the following signals:

- Function Designators
- Function Strobe
- Accumulator Data
- Read Done
- Read Busy
- Busy/done
- Master Clear
- Clock
- Interrupt

Each interface has two 1-bit registers comprising the Done and Busy flags for the channel. The interface can set or clear these flags, and the IOP can sample them through the use of the Read Done and Read Busy control signals.

Function Designators signal

Bits 2^0 through 2^3 of the *f* field of an I/O instruction are used as a function code (0 through 17_8) to an interface (see section 6). This function is interpreted by the interface and can specify different operations to different interfaces. The function code consists of 4 bits sent on lines from the IOP instruction logic to the interface. It precedes any other channel action. The function code is stable on the lines for only 1 CP.

Function Strobe signal

The Function Strobe signal accompanies the function code bits. It alerts the interface to the presence of the function code.

Accumulator Data signal

The use of the Accumulator Data signal depends on the function code and the particular interface. For example, the signal can be treated as a parameter for a Buffer Memory transfer or as a character for a display. Data leaving the accumulator is reliable only for the clock period (CP) containing the Function Strobe signal. The meaning of data coming from the interface to the accumulator also depends on the function code and interface.

Read Done signal

When the Read Done signal is sent to the interface, the Interface Done flag is sent back to the accumulator carry bit. The Done flag is carried on the Busy/done signal line described below.

Read Busy signal

When the Read Busy signal is sent to the interface, the Interface Busy flag is sent back to the accumulator carry bit. The Busy flag is carried on the Busy/done signal line described below.

Busy/done signal

The Busy/done signal responds to the interface's Read Busy or Read Done signals. When the Read Busy signal is received, the Interface Busy flag is copied to the Busy/done signal line. When the Done flag is requested, the Interface Done flag is sent on the Busy/done signal line. The Busy/done flag simply carries the set or cleared state of the requested flag. The signal arrives back at the IOP to enter the carry bit position. It becomes the new carry bit in the same CP that the Read Busy/Done signal issues.

Master Clear signal

A Master Clear signal is sent to each interface when the IOP is deadstarted. After that, the only Master Clear function is an interface interpretation of one of the function codes as a Master Clear command.

Clock signal

The IOP Clock signal is sent to each interface to synchronize the logical operations. The Clock signal is a pulse approximately 12.5 ns wide. The clock speed can be varied for maintenance, in which case the pulse width stays constant. This clock is similar to the clock used in the Cray mainframe; however, the two clock generators are independent.

Interrupt signal

The Interrupt signal from an interface to the IOP causes an interrupt request in the IOP for the channel. An interrupt occurs if the interrupt condition remains present, the Interrupt Enable flag is set for the interface channel, and the System Interrupt Enable is set.

CHANNELS USING A DMA PORT

A channel using a DMA port is an accumulator channel with connections to the Local Memory. In addition to the accumulator channel signals already described, the DMA channel also uses the following signals:

- Local Memory Data
- Local Memory Address
- Request Read
- Request Write
- Acknowledge Write
- Acknowledge Read

Local Memory Data signal

Data is transferred in groups of four 16-bit parcels, 1 parcel per CP in 4 consecutive CPs. A group of 16 lines carries data from Local Memory to the interface. A second group of 16 lines carries interface data to the Local Memory.

When writing data to Local Memory, the first parcel must be sent in the CP after the Acknowledge Write signal is received from the IOP. When reading data from Local Memory, the data is received 7 CPs after the Acknowledge Read signal. In either case, the data is reliable for only 1 CP.

Local Memory Address signal

The Local Memory address signal comes from the interface to the IOP on 14 lines. The interface receives the address from the accumulator under a specific function code, or the interface creates its own address. Any transfer of 4 parcels must have the Local Memory address for the first parcel of the 4-parcel group. The IOP logic then increments the address for the later 3 parcels of the group. The Local Memory address signal must be stable during the CP of the Request Read or the Request Write signals.

Request Read signal

The Request Read signal is sent from the interface to the IOP to take data from Local Memory. This signal accompanies the Local Memory address bits to the IOP.

Request Write signal

The Request Write signal is sent from the interface to the IOP when data is to be sent to Local Memory. This signal accompanies the Local Memory address bits to the IOP.

Acknowledge Read signal

The Acknowledge Read signal is sent from the IOP when the read requested by the interface can be performed. The signal has a minimum delay after the request read of 1 CP, but the delay can be stretched by memory conflicts. It occurs 7 CPs before the first parcel of data of the 4-parcel group and lasts 1 CP.

Acknowledge Write signal

The Acknowledge Write signal is sent from the IOP when the requested write can be performed. The signal has a minimum delay after the Request Write signal of 1 CP, but memory conflicts can increase the delay. The interface must place the first parcel of data on the lines to the IOP in the CP after the Acknowledge Write signal is received at the interface.

READ SEQUENCE

An I/O instruction begins the read sequence by commanding the interface to start a transfer from Local Memory to a peripheral. The interface then begins reading from Local Memory. The address for the transfer is loaded in the interface by an IOP command to that interface. The interface takes that address as the starting address and sends a Request Read signal and the address bits to the IOP. After a minimum delay of 1 CP, the interface receives the Acknowledge Read signal. After 7 CPs, the interface takes the first parcel. It takes the three following parcels in the next 3 CPs. To read another 4 parcels from Local Memory to the interface, another Request Read signal and address must be sent from the interface.

WRITE SEQUENCE

An I/O instruction begins the write sequence by commanding the interface to start a transfer from a peripheral to Local Memory. The interface then begins the write to Local Memory. The address for the transfer is loaded in the interface by an IOP command to that interface. The interface sends a Request Write signal and the address bits to the IOP. After a minimum delay of 1 CP, the interface receives the Acknowledge Write signal from the IOP. The interface must send the first parcel of data to the IOP in the CP after the receipt of the Acknowledge Write signal. Each of the next 3 CPs transfers another parcel into Local Memory.

If another group of 4 parcels is to be written into Local Memory, another Request Write signal and address must be sent from the interface.

STANDARD CHANNELS

Standard functions are assigned to 12 of the 40 available channels and are the same for all IOPs. The interface logic is built into each IOP to handle each standard channel. The functions, channel numbers, and An I/O Processor Macro Language (APML) mnemonics are listed in table 5-1. Six standard channels (006 through 013_8) interconnect the system IOPs as shown in the table. The mnemonics for these channels are shown in relation to the Master I/O Processor (MIOP). The standard channels are all accumulator channels except for one DMA port used by Buffer Memory.

Table 5-1. IOP Standard Channel Assignments
(in relation to the MIOP)

Channel Number	Mnemonic	Purpose/Device
000	IOR	Interrupt request
001	PFR	Program fetch request
002	PXS	Program exit stack
003	LME	Local Memory error
004	RTC	Real-time clock
005	MOS	Buffer Memory interface
006	AIA } Buffer	IOP input
007	AOA } IOP	IOP output
010	AIB } Disk	IOP input
011	AOB } IOP	IOP output
012	AIC } Auxiliary	IOP input
013	AOC } or second Disk IOP	IOP output

Table 5-2 lists the functions recognized by the standard channels. These functions are explained in subsequent paragraphs.

Table 5-2. Standard Channel Functions

Channel	Function	Description
0 (I/O request)	IOR : 10	Read interrupt channel number
1 (Program fetch request)	PFR : 0	Clear the Program Fetch Request flag
	PFR : 6	Clear the Channel Interrupt Enable flag
	PFR : 7	Set the Channel Interrupt Enable flag
	PFR : 10	Read the operand register number

Table 5-2. Standard Channel Functions (continued)

Channel	Function	Description
2 (Program exit stack)	PXS : 0	Clear the Exit Stack Boundary flag
	PXS : 6	Clear the Channel Interrupt Enable flag
	PXS : 7	Set the Channel Interrupt Enable flag
	PXS : 10	Read exit stack pointer, E
	PXS : 11	Read exit stack address, (E)
	PXS : 14	Enter exit stack pointer, E
	PXS : 15	Enter exit stack address, (E)
3 (Local Memory error)	LME : 0	Clear the Local Memory Parity Error flag
	LME : 6	Clear the Channel Interrupt Enable flag
	LME : 7	Set the Channel Interrupt Enable flag
	LME : 10	Read error information
4 (Real-time clock)	RTC : 0	Clear the Channel Done flag
	RTC : 6	Clear the Channel Interrupt Enable flag
	RTC : 7	Set the Channel Interrupt Enable flag
	RTC : 10	Read real-time clock
5 (Buffer Memory)	MOS : 0	Clear the Channel Busy and Done flags
	MOS : 1	Enter the Local Memory address for next transfer
	MOS : 2	Enter upper portion of Buffer Memory address
	MOS : 3	Enter lower portion of Buffer Memory address
	MOS : 4	Read Buffer Memory to Local Memory
	MOS : 5	Write Buffer Memory from Local Memory
	MOS : 6	Clear the Channel Interrupt Enable flag
	MOS : 7	Set the Channel Enable Interrupt flag
	MOS : 14	Set the control flags
6, 10, 12 (IOP input) (AIA, AIB, AIC)	AIA : 0	Clear the Channel Done flag
	AIA : 6	Clear the Channel Interrupt Enable flag
	AIA : 7	Set the Channel Interrupt Enable flag
	AIA : 10	Read input to accumulator and resume channel
7, 11, 13 (IOP output) (AOA, AOB, AOC)	AOA : 0	Clear the Channel Busy and Done flags
	AOA : 1	Enter control bits from accumulator
	AOA : 6	Clear the Channel Interrupt Enable flag
	AOA : 7	Set the Channel Interrupt Enable flag
	AOA : 14	Set the Channel Busy flag and output accumulator data

I/O REQUEST CHANNEL

Channel 0 is reserved for reading interrupt requests. The only function implemented is the following:

Function	Description
IOR : 10	Read interrupt channel number

This function replaces the accumulator contents with the highest priority channel number currently requesting an interrupt. The channel number is loaded into the low-order 9 bits of the accumulator content. The high-order bits are forced to 0. Only the 6 least significant bits are used for the channel number. A zero value means all channel interrupts have been handled.

The interface register used by this channel contains the number of the highest priority channel on which an interrupt is present. The value is changed either by clearing the Interrupt Enable flag for the appropriate channel or by clearing the Done flag for that channel.

For channel 0, the Done flag is always set and the Busy flag is always cleared. Channel 0 can be used with instructions 040 through 043 to set or clear the Carry flag.

PROGRAM FETCH REQUEST CHANNEL

The IOP has an I/O channel (channel 1) that responds to the Program Fetch Request flag when that flag is set. (The Program Fetch Request flag is explained in section 3.) Channel 1 provides a mechanism for calling the IOP monitor program when a new section of program code is required.

A 9-bit interface register holds the operand register number associated with the interrupt request. This register is cleared and a new number entered at the time the Program Fetch Request flag is set.

Function	Description
PFR : 0†	Clear the Program Fetch Request flag. This flag is treated as the Channel Done flag. This channel does not have a Busy flag.
PFR : 6††	Clear the Channel Interrupt Enable flag. The Program Fetch Request flag is not altered in this process.
PFR : 7††	Set the Channel Interrupt Enable flag. The Program Fetch Request flag is not altered in this process.

† Allow 1 CP before checking Busy or Done flags.
†† Allow 1 CP before checking the interrupt channel number (IOR : 10).

Function	Description
PFR : 10	Replace the contents of the accumulator with the contents of the interface register. The interface register contents are loaded into the low-order 9-bit positions in the accumulator. The high-order bits are forced to 0. The contents of the interface register remain unchanged until a new PFR occurs. The Done flag is cleared.

PROGRAM EXIT STACK CHANNEL

An IOP has an I/O channel (channel 2) connected to the program exit stack hardware. Channel 2 provides the monitor program with the information needed to reorganize the contents of the program exit stack when the stack overflows.

Function	Description
PXS : 0†	Clear the Exit Stack Boundary flag. This flag is treated as the Channel Done flag. This channel does not have a Busy flag.
PXS : 6††	Clear the Channel Interrupt Enable flag. The Exit Stack Boundary flag is not altered in this process.
PXS : 7††	Set the Channel Interrupt Enable flag. The Exit Stack Boundary flag is not altered in this process.
PXS : 10	Load the E designator into the low-order 4 bits of the accumulator. The high-order bits of the accumulator are forced to 0.
PXS : 11	Replace the accumulator contents with the contents of the program exit stack address currently pointed to by the E designator.
PXS : 14	Replace the E designator with the low-order 4 bits of the accumulator contents.
PXS : 15	Enter the program exit stack with the accumulator contents at the address currently pointed to by the E designator.

† Allow 1 CP before checking Busy or Done flags.
†† Allow 1 CP before checking the interrupt channel number (IOR : 10).

****

CAUTION

The exit stack is both an I/O device and an integral part of the IOP. Return and exit instructions and interrupts use the exit stack values immediately. Access by the I/O channel to the exit stack takes 4 CPs to complete. Time must be allowed for the I/O channel to complete the transfer before using the exit stack value by a return, exit, or interrupt.

At least 5 CPs must elapse before using any data read from the program stack. Any attempt to use a value changed in the stack within 5 CPs after it is changed has potentially disastrous effects on the program execution sequence. A delay is also necessary when reading from the exit stack or pointer after writing to it.

A simple way to achieve the required delay is to perform 3 circular shifts of 17 positions on the accumulator before exiting the routine. Modifying stack locations or the E pointer should only be done when system interrupts are disabled.

****

Deadstart sequence

On Master Clear, stack location 0 is cleared to a zero value. On the deadstart interrupt, the P register is set to 0 and program execution begins at memory location 0. The exit stack pointer, E, is set to the current value plus 1.

LOCAL MEMORY ERROR CHANNEL

Each IOP has an I/O channel (channel 3) connected to the error detection circuits in Local Memory. Channel 3 provides the error indication in the event of memory malfunction and provides maintenance information. This channel provides information helpful to the maintenance function as quickly as possible. No attempt at continued operation is expected beyond system shutdown. (I/O Subsystem (IOS) error logging on machines with serial numbers of 21 and higher does not replace or interfere with this channel.)

Function	Description
LME : 0	Clear the Local Memory Parity Error flag
LME : 6†	Clear the Channel Interrupt Enable flag
LME : 7†	Set the Channel Interrupt Enable flag
LME : 10	Read error information into the 5 low-order bit positions of the accumulator as follows:

Bit	Meaning
2^0	Bank number 2^0
2^1	Bank number 2^1
2^2	Section number 2^0
2^3	Section number 2^1
2^4	Error bit: 0 Error in 2^0 through 2^7 byte 1 Error in 2^8 through 2^{15} byte

REAL-TIME CLOCK CHANNEL

The IOP real-time clock (RTC) is a 17-bit counter/timer which interrupts the IOP at 1-ms intervals. The RTC increments every CP. Upon reaching a count of 234177_8, it sets the RTC Channel Done flag, clears to 0, and continues incrementing. The RTC channel (channel 4) does not have a Busy flag. (Since the accumulator only holds 16 bits, the low-order bit of the counter is ignored, giving a timing accuracy of 2 CPs, and a count of 116077_8.)

The RTC cannot be set. To time an interval, the RTC must be read at the beginning and end of an interval. Thus, to time an interval, the program must:

- Read the clock at the beginning of the interval
- Store the value in a register
- Read the clock at the end of the interval

This instruction timing adds 8 CPs (4 clock counts) to the measured sequence. The algorithm for determining an interval without RTC interrupts is:

$$\text{Time (ns)}_8 = (\text{RTC ending}_8 - \text{RTC beginning}_8 - 4) \times 2 \text{ (CP ns)}$$

† Allow 1 CP before checking the interrupt channel number (IOR : 10).

If RTC interrupts occur during the interval, the algorithm becomes:

$$\text{Time (ns)}_8 = ((\text{RTC ending}_8 - \text{RTC beginning}_8 - 4) + (116077_8 \times \text{number of interrupts})) \times 2 \text{ (CP ns)}$$

Synchronizing the beginnings of time intervals less than 1 ms with the occurrence of an RTC interrupt eliminates the possibility of an RTC interrupt during the timing sequence.

The functions for the RTC channel follow:

Function	Description
RTC : 0†	Clear the Channel Done flag.
RTC : 6††	Clear the Channel Interrupt Enable flag.
RTC : 7††	Set the Channel Interrupt Enable flag.
RTC : 10	Read the RTC count into the accumulator.

BUFFER MEMORY CHANNEL

The IOP has an I/O channel (channel 5) connected to Buffer Memory. Channel 5 allows the IOP program to transfer blocks of data in either direction between its Local Memory and Buffer Memory. It cannot transfer simultaneously in both directions.

This channel has three interface registers to control the block copy operations. The Buffer Memory address is held in a 24-bit register. The high-order 15 bits of this address are entered with a function 2 request. The low-order 9 bits are entered with a function 3 request. The Local Memory address is held in a 14-bit register which is entered with a function 1 request. The block length in Buffer Memory words is held in a 14-bit register. This register is entered with a function 4 or function 5 request.

The Local Memory address in the channel register is given a value that is a multiple of 4 by forcing the 2 low-order bits of the register content to zero values. This provision allows the maximum data rate to Local Memory.

All three register values are cleared to 0 at the end of a block copy. A zero value is used if a register entry is omitted in the next sequence.

---

† Allow 1 CP before checking Busy or Done flags.
†† Allow 1 CP before checking the interrupt channel number (IOR : 10).

The Busy flag remains set if a multiple-bit error occurred in the transfer at the end of a read transfer (Done flag set).

Function requests for channel 5 are summarized as follows:

Function	Description
MOS : 0†	Clear the Channel Busy and Channel Done flags
MOS : 1	Enter the accumulator contents in the channel interface register for the Local Memory address. The 2 low-order bits are forced to 0.
MOS : 2	Enter the 15 low-order bits of the accumulator contents as the 15 high-order bits of the Buffer Memory address. See figure 5-1.
MOS : 3	Enter the 9 low-order bits of the accumulator contents as the 9 low-order bits of the Buffer Memory address. See figure 5-1.

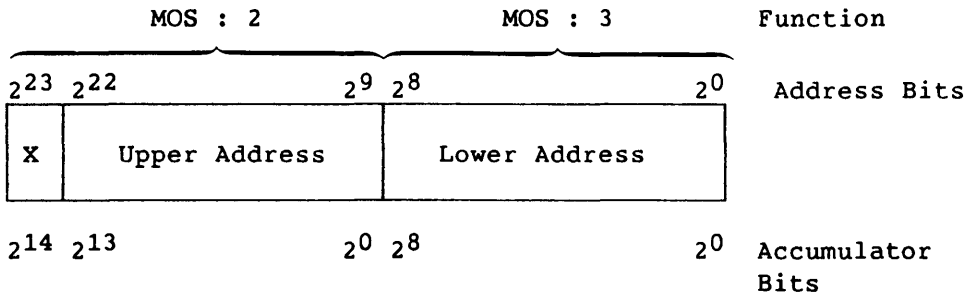


Figure 5-1. Buffer Memory Address Formation

MOS : 4† Initiate the transfer of the block of data from Buffer Memory to Local Memory. The Channel Busy flag is set and the Channel Done flag is cleared by the function request. The 14 low-order bits of the accumulator contents at the time of the function request are entered in the channel interface Block Length register. The Channel Done flag is set and the Channel Busy flag is cleared when the block transfer is complete. If the Block Length register contains a zero value, the block length is set to 65,536 parcels (full Local Memory). The Channel Busy flag remains set if a multiple-bit error occurred in the transfer. Single-bit errors are automatically corrected.

† Allow 1 CP before checking Busy or Done flags.

Function	Description
MOS : 5†	Initiate the transfer of a block of data from Local Memory to Buffer Memory. The Channel Busy flag is set and Channel Done flag is cleared by the function request. The 14 low-order bits of the accumulator contents at the time of the function request are entered in the channel interface Block Length register. The Channel Done flag is set and the Channel Busy flag is cleared when the block transfer is complete.
MOS : 6††	Clear the Channel Interrupt Enable flag
MOS : 7††	Set the Channel Interrupt Enable flag
MOS : 14	Enter the contents of bits 2^1 and 2^2 of the accumulator into the control register. This function is used for diagnostic purposes only.

Bit	Meaning
2^1	Disable write check bits
2^2	Disable refresh

Setting the control bits to 0 enables write check bits or refresh.

Error handling

When an error occurs, the Busy and Done flags are set. A MOS : 0 command must be issued to the channel to clear the Busy and Done flags before another read or write is initiated.

Buffer Memory interface deadstart

The Buffer Memory interface has provisions for deadstarting the IOP. If the IOP is master cleared with the deadstart signal set, the Local Memory Address register, the Buffer Memory Address register, and the Block Length register are cleared. This master clear sets up a 65K parcel transfer.

A MOS : 4 function is initiated when the IOP Master Clear signal goes to 0. The Interrupt Enable flag is set (MOS : 7) and the completed transfer interrupts the IOP.

---

† Allow 1 CP before checking Busy or Done flags.
†† Allow 1 CP before checking the interrupt channel number (IOR : 10).

Buffer Memory interface dead dump

The Buffer Memory interface also provides for a dead dump of the IOP. If the IOP is master cleared with the dead dump signal set, the Local Memory Address, Buffer Memory Address, and the Block Length registers are cleared. A MOS : 5 function is initiated when Master Clear goes to 0. The Interrupt Enable flag is cleared (MOS : 6) and no interrupt occurs when the transfer completes.

I/O PROCESSOR INPUT CHANNELS

The IOP has three input channels (channels 6, 10, and 12) for connection to other IOPs. These channels provide a communication link between IOPs but have no Busy flags. Instead, each channel uses a single 16-bit register to hold the data parcel being transferred. The register is cleared as soon as the data enters the receiving accumulator.

Function	Description
AIA : 0†	Clear the Channel Done flag. This channel does not have a Busy flag.
AIA : 6††	Clear the Channel Interrupt Enable flag.
AIA : 7††	Set the Channel Interrupt Enable flag.
AIA : 10	Read the accumulator data of the other IOP into the accumulator of this IOP and resume the channel. The data does not remain in the interface register after it has been read. This function generates an interrupt on the corresponding output channel in the sending processor.

I/O PROCESSOR OUTPUT CHANNELS

An IOP has three output channels (channels 7, 11, and 13) that connect to other IOPs. These channels provide both a communications link between IOPs and the ability to master clear, deadstart, or dead dump another IOP.

A 4-bit control register receives the 4 low-order bits of the sending IOP accumulator and causes the required action. A 16-bit register holds transmitted data until it is loaded into the receiving accumulator, at which time the data register is cleared.

† Allow 1 CP before checking Busy or Done flags.
†† Allow 1 CP before checking the interrupt channel number (IOR : 10).

Function	Description
AOA : 0†	Clear the Channel Busy and Done flags
AOA : 1	Enter the 4 low-order accumulator bits into the control register. A set bit in the register positions causes the following actions:

Bit	Meaning
2^0	Master clear
2^1	Deadstart
2^2	Dead dump/Master Clear Buffer Memory
2^3	Short transfer

To perform a deadstart from Buffer Memory, set the Master Clear and deadstart control bits simultaneously for at least 200 ns. Then clear both bits. (The deadstart control bit has no effect without the Master Clear control bit.) The deadstart transfer is initiated with a block length of 65K parcels starting at Local Memory address 0 and Buffer Memory address 0. In approximately 2 ms, the transfer completes and interrupts the IOP.

To perform a short deadstart from Buffer Memory, simultaneously set the short transfer bit, deadstart bit, and Master Clear bit for at least 200 ns. Then clear all bits. (The deadstart control bit has no effect without the Master Clear control bit.) When all bits are cleared simultaneously, a transfer of only 4096 parcels begins. This short transfer completes in approximately 100 microseconds and interrupts the IOP.

To perform a dead dump from Local Memory to Buffer Memory, set the Master Clear and dead dump control bits simultaneously for at least 200 ns. Then clear both bits. The dead dump transfer initiates with a block length of 65K parcels, starting with Local Memory address 0 contents going into Buffer Memory address 0. The dead dump completes in approximately 2 ms but does not interrupt the IOP at completion.

† Allow 1 CP before checking the interrupt channel number (IOR : 10).

Function	Description
	To perform a short dead dump from Local Memory to Buffer Memory, simultaneously set the short transfer bit along with the Master Clear and dead dump bits for at least 200 ns. Then clear all bits. (The dead dump control bit has no effect without the Master Clear control bit.) When all bits are cleared simultaneously, the short dead dump transfer of 4096 parcels begins. The transfer takes approximately 100 microseconds to complete and does not interrupt the IOP when complete.
	********** CAUTION ********** Any time the dead dump bit is set while the Master Clear bit is cleared (bit 2^0) on an IOP channel 7, Buffer Memory may be master cleared. This has adverse effects on any ongoing Buffer Memory transfers from an IOP. **********
	Buffer Memory may be master cleared from any IOP channel 7 if the dead dump bit is set while the Master Clear bit (2^0) is cleared. This operation aborts any current Buffer Memory transfers and resynchronizes the error logging channel with Buffer Memory. This operation is normally restricted to diagnostic mode.
AOA : 6†	Clear Channel Interrupt Enable flag
AOA : 7†	Set Channel Interrupt Enable flag
AOA : 14††	Set the Channel Busy flag and output accumulator data. When data is accepted by the receiving IOP, the Busy flag clears and the Done flag sets. This function generates an interrupt on the corresponding input channel in the receiving processor.

† Allow 1 CP before checking the interrupt channel number (IOR : 10).
†† Allow 1 CP before checking Busy or Done flags.

INTERRUPT SEQUENCE

The IOP program is interrupted for service by a monitor program when an interrupt request is present and the System Interrupt Enable flag is set. The System Interrupt Enable flag applies to the entire IOP, as contrasted to the Channel Interrupt Enable flags that apply only to the particular channel. Instruction 003 (I = 1) sets the System Interrupt Enable flag. The interruption occurs upon completion of an instruction in the currently executing program.

A jump or exit instruction must be completed before interrupts are actually set. Therefore, interrupts are not actually enabled until the first nonbranching instruction is executed.

NOTE

A 2-instruction window is necessary for an interrupt to occur.

The interrupt sequence begins with the hardware clearing the System Interrupt Enable flag to prevent further interrupts.

NOTE

The hardware may not always prevent further interrupts. If an I = 1 instruction (enabling system interrupts) is executed with interrupts already enabled, and an interrupt occurs before the next nonbranch instruction, interrupts are reenabled on the first nonbranch instruction in the interrupt handler. To guard against this possibility, the first instruction in the interrupt handler should be I = 0.

The address for the next instruction in the interrupted program is stored in the exit stack. The entry in the exit stack is made by advancing the E register one count and entering the exit stack at the newly pointed position. This sequence also occurs on a return jump execution. Execution of the interrupted program is then suspended and a new program sequence initiated. The address for beginning the monitor program sequence is obtained from the 0 position in the exit stack, without using the E register.

The monitor program, when finished, restores the System Interrupt Enable flag and exits to the interrupted program as if it were a subprogram call. The last two statements in the monitor program should be the equivalent of the following:

```
I = 1    (Set System Interrupt Enable flag)

EXIT     (Exit)
```

When issuing instruction 003 (I = 1), the System Interrupt Enable flag is delayed until the next nonbranch or non-I/O instruction is issued. Instructions 40 through 43 and 70 through 137 do not enable interrupts, allowing the monitor to return to the interruptible activity before an interrupt is accepted.

Instruction 002 (I = 0) should be used at the interrupt handler entrance. If a redundant instruction 003 (I = 1) is executed and an interrupt occurs before a nonbranch or non-I/O instruction is encountered, the interrupt handler is entered (with interrupts disabled). But interrupts are reenabled when the first nonbranch or non-I/O instruction is issued within the interrupt handler.

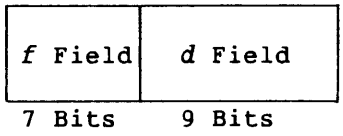
If an I/O channel interrupt is pending and the system interrupt is set, a disable channel interrupt (002) is not performed until that interrupt is processed. The system interrupt occurs 3 CPs after issuing a function 7 to enable the channel interrupt.

IOP INSTRUCTIONS 6

INSTRUCTION FORMAT

Each I/O Processor (IOP) instruction occupies 1 parcel (16 bits) or 2 parcels (32 bits) in Local Memory. A 1-parcel instruction consists of a function code (*f*) field and a designator (*d*) field. A 2-parcel instruction consists of the *f* and *d* fields and a constant (*k*) field as illustrated in figure 6-1.

1-parcel instruction:



2-parcel instruction:

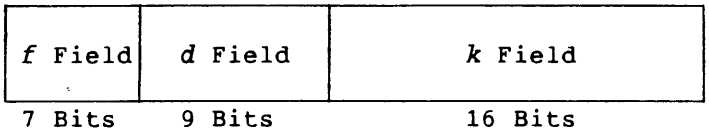


Figure 6-1. Instruction Formats

The 7-bit *f* field, instruction function code, specifies which instruction is to be executed. The 9-bit *d* field designates where the function of the *f* field is to be performed in the machine resources. The *d* field can be thought of as a displacement specification and can be used to:

- Point to a specific operand register when one is required
- Point to a specific I/O channel
- Specify the amount of the displacement of data in a shift instruction
- Specify the amount of displacement forward or backward in program code for a branch instruction
- Function as an operand value

The *d* field is always treated as a 9-bit positive integer. Small integers can be entered directly into the computation from instructions using the *d* field as a constant. Separate instructions are provided to add or subtract this 9-bit constant rather than to consider it as a sign extended quantity. A branch instruction can designate a forward or a backward displacement from the current program location. Separate instructions are provided for the forward and backward jumps using the *d* field as a 9-bit displacement magnitude.

Certain instructions use the program parcel immediately following the instruction as a 16-bit constant field, designated *k*. These instructions can be considered 2-parcel instructions.

INSTRUCTION DESCRIPTIONS

The I/O Processor instructions are described on the following pages. Mnemonics for each instruction are APML statements representing the individual operations performed. The results of each instruction and the steps and clock periods (CPs) involved in executing the instruction are also provided. The use of a - (dash) after a CP *n* means the same conditions hold for that CP as for the previous CP.

Symbols used in the descriptions are as follows:

Symbol	Description
>	Shift right
<	Shift left
>>	Shift right circular
<<	Shift left circular
&	Logical product
#	Not equal to
dd	Contents of operand register specified by *d* field
(*dd*)	Contents of memory location specified by *dd*
iod	3-character channel mnemonic, that is, IOR, PXS,...
B	Contents of register B
(B)	Contents of operand register specified by B register

Refer to appendix A for a summary of I/O Processor instructions.

Instruction 000

PASS

Instruction 000 performs no operation. It can be used to fill program fields with null operations where desired.

CP 0 Issue.

Instruction 001

EXIT

Instruction 001 terminates execution of the current program sequence and returns to the sequence that was suspended by calling this subprogram. The current P register value is discarded. The beginning address for the reinitiated sequence is obtained from the program exit stack at the location currently pointed to by E. The value of E is then decremented by 1. If the value of E was previously 0, decrementing is blocked and the Exit Stack Boundary flag is set. The Exit Stack Boundary flag causes an interrupt of the program sequence for restructuring the contents of the program exit stack.

If the EXIT instruction follows a modification of the program exit stack or the E pointer, at least 4 CPs must elapse between the last modification and the EXIT instruction. Three circular shifts can be used as the necessary delay.

CP 0 Issue.

CP 1 Decrement E.

CP 2 Transmit exit stack data to P register.

Instruction 002

I = 0

Instruction 002 clears the System Interrupt Enable flag.

NOTE

The 002 instruction must be followed by one instruction 000 (PASS). The instruction following the 002 instruction may be skipped if an interrupt occurs while 002 is executing.

CP 0 Issue.

CP 1 Clear System Interrupt Enable flag.

Instruction 003

I = 1

Instruction 003 sets the System Interrupt Enable flag.

When issuing instruction 003, the system interrupt enable is delayed until the next nonbranch or non-I/O instruction is issued. Instructions 040 through 043 and 070 through 137 do not enable interrupts after instruction 003 issues. A delay in setting the flag for this instruction allows the interrupt program to reenable the interrupt mode and then exit to the interrupted program.

If the instruction following instruction 003 is instruction 002, instruction 002 takes precedence and system interrupts are disabled.

CP 0 Issue.

CP 1 -

CP 2 Set System Interrupt Enable flag.

Instruction 004

A = A > *d*

Instruction 004 shifts the contents of the accumulator and the associated Carry flag to the right by *d* bit positions. The Carry flag is the 17th bit and is located to the left of the accumulator contents for this operation. Zero values are entered in the Carry flag and propagated to the right as the shift progresses. No shift is performed if the shift count is 0. The accumulator and Carry flag are cleared if the shift count is greater than 16_{10}.

The low-order 5 bits of the Addend register are interpreted in determining the shift count. High-order bits are ignored.

CP 0 Issue.

CP 1 Transmit *d* to Addend register.

CP 2 Transmit accumulator data and inverted *d* to Shifter; shift mode. The Addend register is available this CP.

CP 3 Transmit Shifter result to accumulator.

Instruction 005

$A = A < d$

Instruction 005 shifts the contents of the accumulator and the associated Carry flag to the left by *d* bit positions. The Carry flag is the 17th bit and is located to the left of the accumulator contents for this operation. Zero values are entered in the low-order bit positions of the accumulator and are propagated to the left as the shift progresses. Bits shifted from the Carry flag are discarded. No shift is performed if the shift count is 0. The accumulator and Carry flag are cleared if the shift count is greater than 16_{10}.

The low-order 5 bits of the Addend register are interpreted in determining the shift count. High-order bits are ignored.

CP 0 Issue.

CP 1 Transmit *d* to Addend register.

CP 2 Transmit accumulator data and *d* to Shifter; shift mode. The Addend register is available this CP.

CP 3 Transmit Shifter result to accumulator.

Instruction 006

A = A >> *d*

Instruction 006 shifts the contents of the accumulator and the associated Carry flag to the right in a circular mode by *d* bit positions. The Carry flag is the 17th bit and is located to the left of the accumulator contents for this operation. No bits are discarded in this shift mode. Bits shifted from the right end of the accumulator are returned to the Carry flag. No shift is performed if the shift count is 0.

The low-order 5 bits of the Addend register are interpreted in determining the shift count. High-order bits are ignored.

CP 0 Issue.

CP 1 Transmit *d* to Addend register.

CP 2 Transmit accumulator data and inverted *d* to Shifter; shift mode. The Addend register is available this CP.

CP 3 Transmit Shifter result to accumulator.

Instruction 007

A = A << *d*

Instruction 007 shifts the contents of the accumulator and the associated Carry flag to the left in a circular mode by *d* bit positions. The Carry flag is the 17th bit and is located to the left of the accumulator contents for this operation. No bits are discarded in this shift mode. Bits shifted from the Carry flag are returned to the low-order bit position in the accumulator. No shift is performed if the shift count is 0.

The low-order 5 bits of the Addend register are interpreted in determining the shift count. High-order bits are ignored.

CP 0 Issue.

CP 1 Transmit *d* to Addend register.

CP 2 Transmit accumulator data and *d* to Shifter; shift mode. The Addend register is available this CP.

CP 3 Transmit Shifter result to accumulator.

Instruction 010

A = *d*

Instruction 010 enters the *d* field into the accumulator as a 9-bit positive integer and clears the Carry flag. The high-order bits are zeros.

CP 0 Issue.

CP 1 Transmit *d* to accumulator.

Instruction 011

A = A & *d*

Instruction 011 forms the bit-by-bit logical product of the previous accumulator contents and the *d* field and places the result in the accumulator. The *d* field is treated as a 9-bit positive integer. It then clears the Carry flag.

The logical product of the previous accumulator contents and the *d* operand from the instruction is formed as the input to the accumulator.

CP 0 Issue.

CP 1 Transmit *d* and accumulator to accumulator.

Instruction 012

A = A + *d*

Instruction 012 adds the *d* field to the previous accumulator contents in the 16-bit twos complement mode and places the result in the accumulator. The *d* field is treated as a 9-bit positive integer in this addition. The instruction complements the Carry flag if a carry is propagated from the accumulator in the addition process.

CP 0 Issue.

CP 1 Transmit *d* to Addend register.

CP 2 Transmit data to Adder; add mode.

CP 3 Transmit Adder result to accumulator.

Instruction 013

A = A - *d*

Instruction 013 subtracts the *d* field from the previous accumulator contents in a 16-bit twos complement mode and places the result in the accumulator. The *d* field is treated as a 9-bit positive integer in this operation. Subtraction is performed by complementing the contents of the Addend register and adding the result to the previous accumulator contents. A 1 is then added to the result. The instruction complements the Carry flag if a carry is propagated from the accumulator during either addition process.

CP 0 Issue.

CP 1 Transmit *d* to Addend register.

CP 2 Transmit data to Adder; subtract mode.

CP 3 Transmit Adder result to accumulator.

Instruction 014

A = *k*

Instruction 014 enters a 16-bit constant in the accumulator and clears the Carry flag. The constant is obtained from the next sequential parcel in the program field. The next instruction is obtained from the following parcel.

CP 0 Issue.

CP 1 Read next parcel out of instruction stack.

CP 2 Transmit *k* data to accumulator. The function of CP 2 is delayed if the next parcel of instruction buffer data is not available in the instruction stack.

Instruction 015

A = A & *k*

Instruction 015 forms the bit-by-bit logical product of the previous accumulator contents and a 16-bit constant and places the result in the accumulator. It then clears the Carry flag. The constant is obtained from the next sequential parcel in the program field. The next instruction is obtained from the following parcel.

The logical product of the previous accumulator contents and the operand constant is formed as the input to the accumulator.

CP 0 Issue.

CP 1 Read the next parcel out of the instruction stack.

CP 2 Transmit *k* data to accumulator. The function of CP 2 is delayed if the next parcel of instruction buffer data is not available in the instruction stack.

Instruction 016

$$A = A + k$$

Instruction 016 adds a 16-bit constant to the previous accumulator contents in a twos complement mode and places the result in the accumulator. The constant is obtained from the next sequential parcel in the program field. The next instruction is obtained from the following parcel. The instruction complements the Carry flag if a carry is propagated from the accumulator in the addition process.

CP 0 Issue.

CP 1 Read next parcel out of the instruction stack.

CP 2 Transmit *k* data to Addend register. The function of CP 2 is delayed if the next parcel of instruction buffer data is not available in the instruction stack.

CP 3 Transmit data to Adder; add mode.

CP 4 Transmit Adder result to accumulator.

Instruction 017

A = A - *k*

Instruction 017 subtracts a 16-bit constant from the previous accumulator contents in a twos complement mode and places the result in the accumulator. The constant is obtained from the next sequential parcel in the program field. The next instruction is obtained from the following parcel. Subtraction is performed by complementing the constant and adding the result to the accumulator contents. A 1 is then added to the result. The instruction complements the Carry flag if a carry is propagated from the accumulator in either addition process.

NOTE

An equivalent function can be performed with instruction 016 using a different constant. Instruction 017 is included in the list for completeness in the translational pattern.

CP 0 Issue.

CP 1 Read next parcel out of the instruction stack.

CP 2 Transmit *k* data to Addend register. The function of CP 2 is delayed if the next parcel of instruction buffer data is not available in the instruction stack.

CP 3 Transmit data to Adder; subtract mode.

CP 4 Transmit Adder result to accumulator.

Instruction 020

A = *dd*

Instruction 020 enters the contents of operand register *d* in the accumulator and clears the Carry flag.

CP 0 Issue.

CP 1 Transmit *d* data to RP register.

CP 2 Transmit operand register data to accumulator.

Instruction 021

A = A & *dd*

Instruction 021 forms the bit-by-bit logical product of the previous accumulator contents and the contents of operand register *d* and places the result in the accumulator. It then clears the Carry flag.

The logical product of the previous accumulator contents and the operand register contents is formed as the input to the accumulator. No additional time is required for this function.

CP 0 Issue.

CP 1 Transmit *d* data to RP register.

CP 2 Transmit operand register data to accumulator.

Instruction 022

A = A + *dd*

Instruction 022 adds the contents of operand register *d* to the previous accumulator contents. Addition is performed in the twos complement mode. The Carry flag is complemented if a carry is propagated from the accumulator in the addition process.

CP 0 Issue.

CP 1 Transmit *d* data to RP register.

CP 2 Transmit operand register data to Addend register.

CP 3 Transmit data to Adder; add mode. The data in the accumulator and Addend register is transmitted to the Adder in this CP. The Addend register is available to accept another operand this CP.

CP 4 Transmit Adder result to accumulator.

Instruction 023

A = A - *dd*

Instruction 023 subtracts the contents of operand register *d* from the previous accumulator contents. Subtraction is performed by complementing the subtrahend and adding the result to the accumulator contents in ones complement mode. A 1 is then added to the result. The instruction complements the Carry flag if a carry is propagated from the accumulator in the addition process.

CP 0 Issue.

CP 1 Transmit *d* data to RP register.

CP 2 Transmit operand register data to Addend register.

CP 3 Transmit data to Adder; subtract mode. The data in the accumulator and Addend register is transmitted to the Adder in this CP. The Addend register is available to accept another operand this CP.

CP 4 Transmit Adder result to accumulator.

Instruction 024

dd = A

Instruction 024 stores the accumulator contents in operand register *d*. This instruction cannot issue if the DP register contains a pointer from a previous instruction.

CP 0 Issue.

CP 1 Transmit *d* data to RP and DP registers.

CP 2 Transmit accumulator data to operand register. The function of CP 2 is delayed if the data is not available in the accumulator during the indicated CP. In this case, the pointers in the RP and DP registers are held until the accumulator data is available.

Instruction 025

dd = A + *dd*

Instruction 025 adds the contents of operand register *d* to the previous accumulator contents. Addition is performed in a twos complement mode. The instruction complements the Carry flag if a carry is propagated from the accumulator in the addition process. It then replaces the contents of operand register *d* with the new accumulator contents.

This instruction cannot issue if the DP register contains a pointer from a previous instruction.

CP 0 Issue.

CP 1 Transmit *d* data to RP and DP registers.

CP 2 Transmit operand register data to Addend register. The RP register pointer is discarded in CP 2 and reentered with the same pointer from DP in CP 4. The function at CP 2 is delayed if the data is not available in the accumulator.

CP 3 Transmit data to Adder; add mode.

CP 4 Transmit Adder result to accumulator. Transmit DP register data to RP register.

CP 5 Transmit accumulator data to operand register.

Instruction 026

dd = *dd* + 1

Instruction 026 replaces the contents of operand register *d* with the previous contents increased by 1. The result is left in the accumulator and in the operand register. The Carry flag is cleared at the beginning of this operation. A 1 is entered in the accumulator. The contents of the operand register enters the Addend register and is then added to the accumulator contents in a twos complement mode. The Carry flag is set if a carry is propagated from the accumulator in the addition process. The result is then returned to the operand register.

This instruction cannot issue if the DP register contains a pointer from a previous instruction.

CP 0 Issue.

CP 1 Transmit *d* data to RP and DP registers.

CP 2 Transmit operand register data to Addend register. Enter a +1 in the accumulator.

The RP register pointer is discarded in CP 2 and reentered with the same pointer from the DP register in CP 4.

CP 3 Transmit data to Adder; add mode.

CP 4 Transmit Adder result to accumulator. Transmit DP register data to RP register.

CP 5 Transmit copy of accumulator data to operand register.

Instruction 027

dd = *dd* - 1

Instruction 027 replaces the contents of operand register *d* with the previous contents decreased by 1. The result is left in the accumulator and in the operand register. The Carry flag is cleared at the beginning of this operation. The accumulator bits are forced set. The contents of the operand register are entered in the Addend register and then added to the accumulator contents. The Carry flag is set if a carry is propagated from the accumulator in the addition process. The result is then returned to the operand register.

This instruction cannot issue if the DP register contains a pointer from a previous instruction.

CP 0 Issue.

CP 1 Transmit *d* data to RP and DP registers.

CP 2 Transmit operand register data to Addend register. Enter a -1 in the accumulator. The RP register pointer is discarded in CP 2 and reentered with the same pointer from the DP register in CP 4.

CP 3 Transmit data to Adder; add mode.

CP 4 Transmit Adder result to accumulator. Transmit DP register data to RP register.

CP 5 Transmit copy of accumulator data to operand register.

Instruction 030

A = (*dd*)

Instruction 030 enters the contents of a Local Memory location in the accumulator. The Local Memory address is obtained from operand register *d*. It then clears the Carry flag.

CP 0 Issue.

CP 1 Transmit *d* data to RP register. The function of CP 1 is delayed if the MA register data from a previous instruction has not been accepted.

CP 2 Transmit operand register data to MA register.

CP 3 Transmit MA register data to bank address registers. Send Read request to memory. The function of CP 3 is repeated until the acceptance signal is received.

CP 4 Acceptance signal from Local Memory.

CP 5 -

CP 6 -

CP 7 Transmit memory data to accumulator.

Instruction 031

A = A & (*dd*)

Instruction 031 forms the bit-by-bit logical product of the previous accumulator contents and the contents of a Local Memory location and places the result in the accumulator. The Local Memory address is obtained from operand register *d*. It then clears the Carry flag.

CP 0 Issue.

CP 1 Transmit *d* data to RP register. The function of CP 1 is delayed if the MA register data from a previous instruction has not been accepted.

CP 2 Transmit operand register data to MA register.

CP 3 Transmit MA register data to bank address registers. Send Read request to memory. The function of CP 3 is repeated until the acceptance signal is received.

CP 4 Acceptance signal from Local Memory.

CP 5 -

CP 6 -

CP 7 Transmit memory data to accumulator. The logical product of memory data and the accumulator contents is done at the input to the accumulator. No additional time is required for this function.

Instruction 032

A = A + (*dd*)

Instruction 032 adds the contents of a Local Memory location to the contents of the accumulator. The Local Memory address is obtained from operand register *d*. The instruction complements the Carry flag if a carry is propagated from the accumulator in the addition process.

CP 0 Issue.

CP 1 Transmit *d* data to RP register. The function of CP 1 is delayed if the MA register data from a previous instruction has not been accepted.

CP 2 Transmit operand register data to MA register.

CP 3 Transmit MA register data to bank address registers. Send Read request to memory. The function of CP 3 is repeated until the acceptance signal is received.

CP 4 Acceptance signal from Local Memory.

CP 5 -

CP 6 -

CP 7 Transmit memory data to Addend register.

CP 8 Transmit data to Adder; add mode. The data from the accumulator and Addend register is transmitted to the Adder in CP 8. The Addend register is available this CP to accept another operand.

CP 9 Transmit Adder result to accumulator.

Instruction 033

A = A - (*dd*)

Instruction 033 subtracts the contents of a Local Memory location from the contents of the accumulator. The Local Memory address is obtained from operand register *d*. Subtraction is performed by complementing the subtrahend and adding the result to the accumulator contents in a ones complement mode. A 1 is then added to the result. The instruction complements the Carry flag if a carry is propagated from the accumulator during either addition process.

CP 0 Issue.

CP 1 Transmit *d* data to RP register. The function of CP 1 is delayed if the MA register data from a previous instruction has not been accepted.

CP 2 Transmit operand register data to MA register.

CP 3 Transmit MA register data to bank address registers. Send Read request to memory. The function of CP 3 is repeated until the acceptance signal is received.

CP 4 Acceptance signal from Local Memory.

CP 5 -

CP 6 -

CP 7 Transmit memory data to Addend register.

CP 8 Transmit data to Adder; subtract mode. The data from the accumulator and Addend register is transmitted to the Adder in CP 8. The Addend register is available this CP to accept another operand.

The complementing of the Addend data is accomplished in the transmission of the data from the Addend register to the Adder. The addition of +1 to the result is accomplished in this same CP in the Adder.

CP 9 Transmit Adder result to accumulator.

Instruction 034

(*dd*) = A

Instruction 034 replaces the contents of a Local Memory location with the current contents of the accumulator. The Local Memory address is obtained from operand register *d*.

CP 0 Issue.

CP 1 Transmit *d* data to RP register. The function of CP 1 is delayed if the MA register data from a previous instruction has not been accepted or the accumulator data is not available.

CP 2 Transmit operand register data to MA register.

CP 3 Transmit MA register data to bank address registers. Transmit a copy of the accumulator data to bank operand registers. Send Write request to memory. The function of CP 3 is repeated until an acceptance signal is received.

CP 4 Acceptance signal from Local Memory.

CP 5 -

CP 6 Write complete.

Instruction 035

(*dd*) = A + (*dd*)

Instruction 035 replaces the contents of a Local Memory location with its previous contents plus the current accumulator contents. The Local Memory address is obtained from operand register *d*. Addition is performed using the accumulator in a 16-bit twos complement mode. The Carry flag is complemented if a carry is propagated from the accumulator in the addition process. The result is left in the accumulator as well as transmitted to the Local Memory location.

CP 0 Issue.

CP 1 Transmit *d* data to RP register. The function of CP 1 is delayed if the MA register data from a previous instruction has not been accepted or the accumulator data is not available.

CP 2 Transmit operand register data to MA register.

CP 3 Transmit MA register data to bank address registers. Send Read request to memory. The function of CP 3 is repeated until an acceptance signal is received.

CP 4 Acceptance signal from Local Memory. The MA register data is held from CP 4.

CP 5 -

CP 6 -

CP 7 Transmit memory data to Addend register.

CP 8 Transmit data to Adder; add mode.

CP 9 Transmit Adder result to accumulator.

CP 10 Transmit MA register data to bank address registers. Transmit a copy of the accumulator data to bank operand registers. Send Write request to memory. The function of CP 10 is repeated until an acceptance signal is received.

CP 11 Acceptance signal from Local Memory. The accumulator is available this CP for next instruction.

CP 12 -

CP 13 Write complete.

Instruction 036

$(dd) = (dd) + 1$

Instruction 036 increments the contents of a Local Memory location by 1. The Local Memory address is obtained from operand register *d*. This operation is performed using the accumulator. The accumulator and Carry flag are cleared. A +1 is entered in the accumulator. The contents of the memory location is entered in the Addend register and then added to the accumulator contents. The result is returned to Local Memory. The result remains in the accumulator. The Carry flag is complemented if a carry is propagated from the accumulator in the addition process.

CP 0 Issue.

CP 1 Transmit *d* data to RP register. The function of CP 1 is delayed if the MA register data from a previous instruction has not been accepted.

CP 2 Transmit operand register data to MA register.

CP 3 Transmit MA register data to bank address registers. Send Read request to memory. The function of CP 3 is repeated until an acceptance signal is received.

CP 4 Acceptance signal from Local Memory. The MA register data is held from CP 4.

CP 5 -

CP 6 -

CP 7 Transmit memory data to Addend register. Clear Carry flag and enter +1 in accumulator.

CP 8 Transmit data to Adder; add mode.

CP 9 Transmit Adder result to accumulator.

CP 10 Transmit MA register data to bank address registers. Transmit copy of accumulator data to bank operand registers. Send Write request to memory. The function of CP 10 is repeated until the acceptance signal is received.

CP 11 Acceptance signal from Local Memory. The accumulator is available this CP for next instruction.

CP 12 -

CP 13 Write complete.

Instruction 037

(*dd*) = (*dd*) - 1

Instruction 037 decrements the contents of a Local Memory location by 1. The Local Memory address is obtained from operand register *d*. This operation is performed using the accumulator. The Carry flag is cleared. The accumulator bits are forced set. The contents of the Local Memory location are entered in the Addend register and then added to the accumulator contents. The result is returned to the Local Memory location. The result also remains in the accumulator. The Carry flag is complemented if a carry is propagated from the accumulator in the addition process.

CP 0 Issue.

CP 1 Transmit *d* data to RP register. The function of CP 1 is delayed if the MA register data from a previous instruction has not been accepted.

CP 2 Transmit operand register data to MA register.

CP 3 Transmit MA register data to bank address registers. Send read request to memory. The function of CP 3 is repeated until an acceptance signal is received.

CP 4 Acceptance signal from Local Memory. The MA register data is held from CP 4.

CP 5 -

CP 6 -

CP 7 Transmit memory data to Addend register. Clear Carry flag and enter all ones in accumulator.

CP 8 Transmit data to Adder; add mode.

CP 9 Transmit Adder result to accumulator.

CP 10 Transmit MA register data to bank address registers. Send Write request to memory. Transmit copy of accumulator data to bank operand registers. The function of CP 10 is repeated until the acceptance signal is received.

CP 11 Acceptance signal from Local Memory. The accumulator is available this CP for next instruction.

CP 12 -

CP 13 Write complete.

Instruction 040

C = 1, *iod* = DN

Instruction 040 forces the Carry flag to the same state as the channel *d* Done flag.

No other instruction can issue until CP 5. A delay of 1 CP must be inserted if an I/O instruction is issued that alters the Busy or Done flags before instruction 040.

Channel 000 is always done. The Carry flag can be set by setting *d* = 000 in this instruction.

CP 0 Issue.

CP 1 Transmit *d* field to I/O channels.

CP 2 -

CP 3 -

CP 4 -

CP 5 Force state of Carry flag.

Instruction 041

C = 1, *iod* = BZ

Instruction 041 forces the Carry flag to the same state as the channel *d* Busy flag.

No other instruction can issue until CP 5. A delay of 1 CP must be inserted if an I/O instruction is issued that alters the Busy or Done flags before instruction 41.

Channel 000 is never busy. The Carry flag can be forced clear by setting *d* to 000 in this instruction.

CP 0 Issue.

CP 1 Transmit *d* field to I/O channels.

CP 2 -

CP 3 -

CP 4 -

CP 5 Force state of Carry flag.

Instruction 042

C = 1, IOB = DN

Instruction 042 forces the Carry flag to the same state as the Done flag of the channel specified by the contents of the B register.

No other instruction can issue until CP 5. A delay of 1 CP must be inserted if an I/O instruction is issued that alters the Busy or Done flags before instruction 42.

This instruction cannot issue if the B register is reserved.

Channel 000 is always done. The Carry flag can be forced clear by setting B to 000 in this instruction.

CP 0 Issue.

CP 1 Transmit B designator to I/O channels.

CP 2 -

CP 3 -

CP 4 -

CP 5 Force state of Carry flag.

Instruction 043

C = 1, IOB = BZ

Instruction 043 forces the Carry flag to the same state as the Busy flag of the channel specified by the contents of the B register.

No other instruction can issue until CP 5. A delay of 1 CP must be inserted if an I/O instruction is issued that alters the Busy or Done flags before instruction 43.

This instruction cannot issue if the B register is reserved.

Channel 000 is never busy. The Carry flag can be forced clear by setting B to 000 for use by this instruction.

CP 0 Issue.

CP 1 Transmit B designator to I/O channels.

CP 2 -

CP 3 -

CP 4 -

CP 5 Force state of Carry flag.

Instruction 044

A = A > B

Instruction 044 shifts the contents of the accumulator and the associated Carry flag to the right by B bit positions. The Carry flag is the 17th bit to the left of the accumulator contents for this operation. Zero values are entered in the Carry flag and propagated to the right as the shift progresses. No shift is performed if the shift count is 0. The accumulator and Carry flag are cleared if the shift count is greater than or equal to 17_{10}.

The low-order 5 bits of the Addend register contents are interpreted in determining the shift count. High-order bits are ignored.

CP 0 Issue.

CP 1 Transmit B to Addend register.

CP 2 Transmit data and inverted B to Shifter; shift mode. The Addend register is available this CP.

CP 3 Transmit Shifter result to accumulator.

Instruction 045

A = A < B

Instruction 045 shifts the contents of the accumulator and the associated Carry flag to the left by B bit positions. The Carry flag is the 17th bit to the left of the accumulator contents for this operation. Zero values are entered in the low-order bit positions of the accumulator and are propagated to the left as the shift progresses. Bits shifted from the Carry flag are discarded. No shift is performed if the shift count is 0. The accumulator and Carry flag are cleared if the shift count is greater than or equal to 17_{10}.

The low-order 5 bits of the Addend register contents are interpreted in determining the shift count. High-order bits are ignored.

CP 0 Issue.

CP 1 Transmit B to Addend register.

CP 2 Transmit data and B to Shifter; shift mode. The Addend register is available this CP.

CP 3 Transmit Shifter result to accumulator.

Instruction 046

A = A >> B

Instruction 046 shifts the contents of the accumulator and the associated Carry flag to the right in a circular mode by B bit positions. The Carry flag is the 17th bit to the left of the accumulator contents for this operation. No bits are discarded in this shift mode. Bits shifted from the right end of the accumulator are returned to the Carry flag. No shift is performed if the shift count is 0.

The low-order 5 bits of the Addend register contents are interpreted in determining the shift count. High-order bits are ignored.

CP 0 Issue.

CP 1 Transmit B to Addend register.

CP 2 Transmit data and inverted B to Shifter; shift mode. The Addend register is available this CP.

CP 3 Transmit Shifter result to accumulator.

Instruction 047

A = A << B

Instruction 047 shifts the contents of the accumulator and the associated Carry flag to the left in a circular mode by B bit positions. The Carry flag is the 17th bit to the left of the accumulator contents for this operation. No bits are discarded in this shift mode. Bits shifted from the Carry flag of the accumulator are returned to the right end of the accumulator. No shift is performed if the shift count is 0.

The low-order 5 bits of the Addend register contents are interpreted in determining the shift count. High-order bits are ignored.

CP 0 Issue.

CP 1 Transmit B to Addend register.

CP 2 Transmit data and B to Shifter; shift mode. The Addend register is available this CP.

CP 3 Transmit Shifter result to accumulator.

Instruction 050

A = B

Instruction 050 enters the B register contents in the accumulator as a 9-bit positive integer and clears the Carry flag. The high-order bits are 0.

CP 0 Issue.

CP 1 Transmit B to accumulator.

Instruction 051

A = A & B

Instruction 051 forms the bit-by-bit logical product of the previous accumulator contents and the B register contents and places it in the accumulator. The B register contents are treated as a 9-bit positive integer. The instruction clears the Carry flag.

The logical product of the previous accumulator contents and the operand from the instruction are formed at the input to the accumulator.

CP 0 Issue.

CP 1 Transmit B and accumulator to accumulator.

Instruction 052

A = A + B

Instruction 052 adds the B register contents to the previous accumulator contents in the 16-bit twos complement mode. The B register contents are treated as a 9-bit positive integer in this addition. The instruction complements the Carry flag if a carry is propagated from the accumulator in the addition operation.

CP 0 Issue.

CP 1 Transmit B to Addend register.

CP 2 Transmit data to Adder; add mode.

CP 3 Transmit Adder result to accumulator.

Instruction 053

A = A - B

Instruction 053 subtracts the B register contents from the previous accumulator contents in a 16-bit twos complement mode. The B register contents are treated as a 9-bit positive integer in this operation. Subtraction is performed by complementing the contents of the Addend register and adding the result to the previous accumulator contents. A 1 is then added to the result. The instruction complements the Carry flag if a carry is propagated from the accumulator during either addition.

CP 0 Issue.

CP 1 Transmit B to Addend register.

CP 2 Transmit data to Adder; subtract mode.

CP 3 Transmit Adder result to accumulator.

Instruction 054

B = A

Instruction 054 replaces the B register contents with the low-order 9 bits of the accumulator.

This instruction cannot issue until the accumulator sequence has been completed for any previous instructions.

CP 0 Issue.

CP 1 -

CP 2 - (Accumulator is available this CP.)

CP 3 Enter B with accumulator data.

Instruction 055

B = A + B

Instruction 055 adds the contents of the B register to the previous accumulator contents. The B register contents are treated as a 9-bit positive integer in this operation. Addition is performed in a 16-bit twos complement mode. The instruction complements the Carry flag if a carry is propagated from the accumulator in the addition process and then replaces the B register contents with the low-order 9 bits of the accumulator contents.

CP 0 Issue.

CP 1 Transmit B to Addend register.

CP 2 Transmit data to Adder; add mode.

CP 3 Transmit Adder result to accumulator. The Addend register is available this CP.

CP 4 Transmit a copy of the accumulator data to B register.

Instruction 056

B = B + 1

Instruction 056 replaces the contents of the B register with its previous contents increased by 1. The result is left in the accumulator as well as in the B register. The Carry flag is cleared at the beginning of this operation. A 1 is entered in the accumulator. The contents of the B register are then added to the accumulator contents in a 16-bit twos complement mode. The B register contents are treated as a 9-bit positive integer in this process. The low-order 9 bits of the accumulator contents are then returned to the B register.

CP 0 Issue.

CP 1 Transmit B to Addend register. Enter a +1 in the accumulator.

CP 2 Transmit data to Adder; add mode. The Addend register is available this CP.

CP 3 Transmit Adder result to accumulator.

CP 4 Transmit a copy of the accumulator data to B register.

Instruction 057

B = B - 1

Instruction 057 replaces the contents of the B register with its previous contents decreased by 1. The result is left in the accumulator and in the B register. The Carry flag is cleared at the beginning of this operation. The accumulator bits are forced set. The contents of the operand register are entered in the Addend register and then added to the accumulator contents. The B register contents are treated as a 9-bit positive integer in this process. The Carry flag is complemented if a carry is propagated from the accumulator in the addition process. The low-order 9 bits of the accumulator contents are then returned to the B register.

CP 0 Issue.

CP 1 Transmit B to Addend register. Enter a -1 in the accumulator.

CP 2 Transmit accumulator data to Adder; add mode. The Addend register is available this CP.

CP 3 Transmit Adder result to accumulator.

CP 4 Transmit a copy of the accumulator data to B register.

Instruction 060

A = (B)

Instruction 060 enters the contents of operand register B in the accumulator. It then clears the Carry flag.

CP 0 Issue.

CP 1 Transmit B data to RP register.

CP 2 Transmit operand register data to accumulator.

Instruction 061

A = A & (B)

Instruction 061 forms the bit-by-bit logical product of the previous accumulator contents and the contents of operand register B and places the result in the accumulator. It then clears the Carry flag.

The logical product of the previous accumulator contents and the operand register contents is formed at the input to the accumulator. No additional time is required for this function.

CP 0 Issue.

CP 1 Transmit B data to RP register.

CP 2 Transmit operand register data to accumulator.

Instruction 062

A = A + (B)

Instruction 062 adds the contents of operand register B to the previous accumulator contents. Addition is performed in a twos complement mode. The instruction complements the Carry flag if a carry is propagated from the accumulator in the addition process.

CP 0 Issue.

CP 1 Transmit B data to RP register.

CP 2 Transmit operand register data to Addend register.

CP 3 Transmit data to Adder; add mode. The data in the accumulator and Addend register is transmitted to the Adder in CP 3. The Addend register is available this CP to accept another operand.

CP 4 Transmit Adder result to accumulator.

Instruction 063

A = A - (B)

Instruction 063 subtracts the contents of operand register B from the previous accumulator contents. Subtraction is performed by complementing the subtrahend and adding the result to the accumulator contents in a twos complement mode. A 1 is then added to the result. The instruction complements the Carry flag if a carry is propagated from the accumulator during either addition process.

CP 0 Issue.

CP 1 Transmit B data to RP register.

CP 2 Transmit operand register data to Addend register.

CP 3 Transmit data to Adder; subtract mode. The data in the accumulator and Addend register is transmitted to the Adder in CP 3. The Addend register is available this CP to accept another operand.

CP 4 Transmit Adder result to accumulator.

Instruction 064

(B) = A

Instruction 064 stores the accumulator contents in operand register B.

This instruction cannot issue if the DP register contains a pointer from a previous instruction.

CP 0 Issue.

CP 1 Transmit B data to RP and DP registers.

CP 2 Transmit accumulator data to operand register. The function of CP 2 is delayed if the data is not available in the accumulator during the indicated CP. In this case, the pointers in RP and DP registers are held until the accumulator data is available.

Instruction 065

(B) = A + (B)

Instruction 065 adds the contents of operand register B to the contents of the accumulator. Addition is performed in a twos complement mode. The instruction complements the Carry flag if a carry is propagated from the accumulator in the addition process. The instruction then replaces the contents of operand register B with the new accumulator contents.

This instruction cannot issue if the DP register contains a pointer from a previous instruction.

CP 0 Issue.

CP 1 Transmit B data to RP and DP registers.

CP 2 Transmit operand register data to Addend register. The RP register pointer is discarded in CP 2 and reentered with the same pointer from the DP register in CP 4.

CP 3 Transmit data to Adder; add mode. The function at CP 3 is delayed if the data is not available in the accumulator. The Addend register is available this CP.

CP 4 Transmit Adder result to accumulator. Transmit DP register data to RP register.

CP 5 Transmit a copy of the accumulator data to operand register.

Instruction 066

(B) = (B) + 1

Instruction 066 replaces the contents of operand register B with its previous contents increased by 1. The result is left in the accumulator and in the operand register. The Carry flag is cleared at the beginning of this operation. A 1 is entered in the accumulator. The contents of the operand register are entered in the Addend register and are then added to the accumulator contents in a twos complement mode. The Carry flag is complemented if a carry is propagated from the accumulator in the addition process. The result is then returned to the operand register.

This instruction cannot issue if the DP register contains a pointer from a previous instruction.

CP 0 Issue.

CP 1 Transmit B data to RP and DP registers.

CP 2 Transmit operand register data to Addend register. Enter a +1 in the accumulator. The RP register pointer is discarded in CP 2 and reentered with the same pointer from the DP register in CP 4.

CP 3 Transmit data to Adder; add mode. The Addend register is available this CP.

CP 4 Transmit Adder result to accumulator. Transmit DP register data to RP register.

CP 5 Transmit copy of accumulator data to operand register.

Instruction 067

(B) = (B) - 1

Instruction 067 replaces the contents of operand register B with its previous contents decreased by 1. The result is left in the accumulator as well as in the operand register. The Carry flag is cleared at the beginning of this operation. The accumulator bits are forced set. The contents of the operand register are entered in the Addend register and then added to the accumulator contents. The Carry flag is complemented if a carry is propagated from the accumulator in the addition process. The result is then returned to the operand register.

This instruction cannot issue if the DP register contains a pointer from a previous instruction.

CP 0 Issue.

CP 1 Transmit B data to RP and DP registers.

CP 2 Transmit operand register data to Addend register. Enter an all ones value in the accumulator. The RP register pointer is discarded in CP 2 and reentered with the same pointer from the DP register in CP 4.

CP 3 Transmit data to Adder; add mode. The Addend register is available this CP.

CP 4 Transmit Adder result to accumulator. Transmit DP register data to RP register.

CP 5 Transmit copy of accumulator data to operand register.

Instruction 070

P = P + *d*

Instruction 070 terminates the current program sequence and begins a new sequence. The initial address for the new sequence is obtained by adding the *d* field to the address of the current instruction. The *d* field is treated as a 9-bit positive integer and is added in a 16-bit twos complement mode. The accumulator contents and Carry flag are not altered in this process.

The issue of further instruction is blocked until the branch mode is resolved.

CP 0 Issue.

CP 1 Transmit P data to background accumulator. Transmit *d* data to background Addend register.

CP 2 Transmit background accumulator data to Adder; add mode.

CP 3 Transmit Adder result to P register. The first instruction in the new program sequence is transmitted from the instruction stack to the II register in CP 3 if the branch is within the range of the stack. That instruction can issue at CP 5.

CP 4 If the branch is out of stack, a fetch request at the new P address is generated. This action can be delayed *m* CPs due to a previous internal fetch request.

CP *m*+1 Memory request generated by the fetch. A delay of *n* CPs is possible due to memory conflicts.

CP *n*+1 Acceptance signal from memory.

CP *n*+2 Transmit memory data to II register.

CP *n*+3 II data available for decode.

CP *n*+4 Issue new instruction.

Instruction 071

P = P - *d*

Instruction 071 terminates the current program sequence and begins a new sequence. The initial address for the new sequence is obtained by subtracting the *d* field from the address of the current instruction. The *d* field is treated as a 9-bit positive integer and is subtracted in a 16-bit twos complement mode. The accumulator contents and Carry flag are not altered in this process.

The issue of further instruction is blocked until the branch mode is resolved.

CP 0 Issue.

CP 1 Transmit P data to background accumulator. Transmit *d* to background Addend register.

CP 2 Transmit background accumulator data to Adder; subtract mode.

CP 3 Transmit Adder result to P register. The first instruction in the new program sequence is transmitted from the instruction stack to the II register in CP 3 if the branch is within the range of the stack. This instruction can issue at CP 5.

CP 4 If the branch is out of stack, a fetch request at the new P address is generated. This action may be delayed *m* CPs due to previous internal fetch requests.

CP *m*+1 Memory request generated by the fetch. A delay of *n* CPs is possible due to memory conflicts.

CP *n*+1 Acceptance signal from memory.

CP *n*+2 Transmit memory data to II register.

CP *n*+3 II data available for decode.

CP *n*+4 Issue new instruction.

Instruction 072

R = P + *d*

Instruction 072 suspends execution of the current program sequence and calls a subprogram for execution by the following actions. It advances the value of E by 1 and stores the address of the next sequential instruction of this program sequence in the program exit stack. Then it begins executing a new program sequence. The initial address for the new sequence is obtained by adding the *d* field to the address of the current instruction. The *d* field is treated as a 9-bit positive integer and is added in a 16-bit twos complement mode. The accumulator contents and Carry flag are not altered in this process.

CP 0 Issue.

CP 1 Transmit P data to background accumulator. Transmit *d* to background Addend register.

CP 2 Transmit background accumulator data to Adder; add mode. Advance E by 1.

CP 3 Transmit P+1 data to program exit stack. Transmit Adder result to P register.

CP 4 If the branch is out of stack, a fetch request at the new P address is generated. This action can be delayed *m* CPs due to a previous internal fetch request.

CP *m*+1 Memory request generated by the fetch. A delay of *n* CPs is possible due to memory conflicts.

CP *n*+1 Acceptance signal from memory.

CP *n*+2 Transmit memory data to II register.

CP *n*+3 II data available for decode.

CP *n*+4 Issue new instruction.

Instruction 073

R = P - *d*

Instruction 073 suspends execution of the current program sequence and calls a subprogram for execution by the following actions. It advances the value of E by 1 and stores the address of the next sequential instruction of this program sequence in the program exit stack. Then it begins executing a new program sequence. The initial address for the new sequence is obtained by subtracting the *d* field from the address of the current instruction. The *d* field is treated as a 9-bit positive integer and is subtracted in a 16-bit twos complement mode. The accumulator contents and Carry flag are not altered in this process.

CP 0 Issue.

CP 1 Transmit P data to background accumulator. Transmit *d* to background Addend register.

CP 2 Transmit background accumulator data to Adder; subtract mode. Advance E by 1.

CP 3 Transmit P+1 data to program exit stack. Transmit Adder result to P register.

CP 4 If the branch is out of stack, a fetch request at the new P address is generated. This action can be delayed *m* CPs due to a previous internal fetch request.

CP *m*+1 Memory request generated by the fetch. A delay of *n* CPs is possible due to memory conflicts.

CP *n*+1 Acceptance signal from memory.

CP *n*+2 Transmit memory data to II register.

CP *n*+3 II data available for decode.

CP *n*+4 Issue new instruction.

Instruction 074

P = *dd*

Instruction 074 terminates the current program sequence and begins a new sequence. The initial address for the new sequence is obtained from operand register *d*.

The Program Fetch Request flag is set if the contents of operand register *d* are 0. The issue of further instructions is blocked until the first instruction of the new sequence is in the II register.

CP 0 Issue.

CP 1 Transmit *d* data to RP register.

CP 2 Transmit operand register data to background accumulator. Enter 0 in the background Addend register.

CP 3 Transmit background accumulator data to Adder; add mode.

CP 4 Transmit Adder result to P register. A fetch request at the new P address is generated. This action can be delayed *m* CPs due to a previous internal fetch request.

CP *m*+1 Memory request generated by fetch. A delay of *n* CPs is possible due to memory conflicts.

CP *n*+1 Acceptance signal from memory.

CP *n*+2 Transmit memory data to II register.

CP *n*+3 II data available for decode.

CP *n*+4 Issue new instruction.

Instruction 075

P = *dd* + *k*

Instruction 075 terminates the current program sequence and begins a new sequence. The initial address for the new sequence is obtained by adding the contents of operand register *d* to the next parcel of the current program sequence. Addition is performed in a 16-bit twos complement mode. The contents of the accumulator and Carry flag are not altered in this process.

The Program Fetch Request flag is set if the content of operand register *d* is 0. The issue of further instructions is blocked until the first instruction of the new program sequence is not in the II register in CP 1.

CP 0 Issue.

CP 1 Transmit *d* data to RP register.

CP 2 Transmit operand register data to background accumulator. Transmit *k* data to background Addend register.

CP 3 Transmit background accumulator data to Adder; add mode.

CP 4 Transmit Adder result to P register. A fetch request at the new P address is generated. This action can be delayed *m* CPs due to a previous internal fetch request.

CP *m*+1 Memory request generated by the fetch. A delay of *n* CPs is possible due to memory conflicts.

CP *n*+1 Acceptance signal from memory.

CP *n*+2 Transmit memory data to II register.

CP *n*+3 II data available for decode.

CP *n*+4 Issue new instruction.

Instruction 076

R = *dd*

Instruction 076 suspends execution of the current program sequence and calls a subprogram for execution by the following actions. It advances the value of E by 1, stores the address of the next sequential instruction of this sequence in the program exit stack, and begins executing a new program sequence. The initial address for the new sequence is obtained from operand register *d*.

The Exit Stack Boundary flag is set if the advanced E value is 14. The Program Fetch Request flag is set if the contents of operand register *d* are 0. Issuing further instructions is blocked until the first instruction of the new program sequence is in the II register.

CP 0 Issue.

CP 1 Transmit *d* data to RP register.

CP 2 Transmit operand register data to background accumulator. Enter a 0 value in background Addend register.

CP 3 Transmit background accumulator data to Adder; add mode. Advance E by 1.

CP 4 Transmit P+1 data to exit stack. Transmit Adder result to P register. A fetch request at the new P address is generated. This action can be delayed *m* CPs due to a previous internal fetch request.

CP *m*+1 Memory request generated by the fetch. A delay of *n* CPs is possible due to memory conflicts.

CP *n*+1 Acceptance signal from memory.

CP *n*+2 Transmit memory data to II register.

CP *n*+3 II data available for decode.

CP *n*+4 Issue new instruction.

Instruction 077

R = *dd* + *k*

Instruction 077 suspends execution of the current program sequence and calls a subprogram for execution by the following actions. It advances the value of E by 1 and stores the address that is 2 greater than the address of the current instruction in the program exit stack. Then it begins a new program sequence. The initial address for the new sequence is obtained by adding the contents of operand register *d* to the next parcel of the current program sequence.

Addition is performed in a 16-bit twos complement mode. The contents of the accumulator and Carry flag are not altered in this process.

The Exit Stack Boundary flag is set if the advanced E value is 14. The Program Fetch Request flag is set if the contents of operand register *d* are 0. The issue of further instructions is blocked until the first instruction of the new program sequence is in the II register.

CP 0 Issue.

CP 1 Transmit *d* data to RP register.

CP 2 Transmit operand register data to background accumulator. Transmit *k* data to background Addend register. Entry of the *k* data in the background Addend register is delayed if the next parcel of the program sequence is not in the II register in CP 2.

CP 3 Transmit background accumulator data to Adder; add mode. Advance E by 1.

CP 4 Transmit P+2 data to exit stack. Transmit Adder result to P register. A fetch request at the new P address is generated. This action can be delayed *m* CPs due to a previous internal fetch request.

CP *m*+1 Memory request generated by the fetch. A delay of *n* CPs is possible due to memory conflicts.

CP *n*+1 Acceptance signal from memory.

CP *n*+2 Transmit memory data to II register.

CP *n*+3 II data available for decode.

CP *n*+4 Issue new instruction.

Conditional Branch Instructions 100 through 137

Instructions 100 through 137 are branch instructions that 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 these eight modes are combined with four branch criteria to form the set of instructions 100 through 137. The branch criteria are as follows:

(Any branch instruction 070 through 077)------, C = 0
(Any branch instruction 070 through 077)------, C # 0
(Any branch instruction 070 through 077)------, A = 0
(Any branch instruction 070 through 077)------, A # 0

The C = 0 branch condition causes the branch to be taken if the Carry flag is 0. If the Carry flag is set, the current program sequence is continued.

The C # 0 branch condition is the complement of C = 0. The branch is taken if the Carry flag is set. The current sequence is continued if the Carry flag is 0.

The A = 0 branch condition causes the branch to be taken if the accumulator contents are 0. If the accumulator contents are nonzero, the current sequence is continued.

The A # 0 branch condition is the complement of A = 0. The branch is taken if the accumulator contents are nonzero. The current sequence is continued if the accumulator contents are 0.

If the branch is taken, the timing of the above sequences is the same as the timing of the corresponding instruction in the unconditional mode. Formation of the branch condition requires 1 CP after the accumulator has received the desired data. The issue of the next instruction is delayed until the branch criterion is available. If the branch criterion is available in CP 0 and the branch is not taken based on that criterion, the next instruction in the current program sequence can issue in the next CP.

Conditional branch instructions 100 through 137 follow.

Instruction IOP	APML	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

Instruction IOP	APML	Description
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 *d* if carry = 0
121	P = *dd*, C # 0	Jump to address in operand register *d* if carry ≠ 0
122	P = *dd*, A = 0	Jump to address in operand register *d* if A = 0
123	P = *dd*, A # 0	Jump to address in operand register *d* if A ≠ 0
124	P = *dd* + *k*, C = 0	Jump to address in operand register *d* + *k* if carry = 0
125	P = *dd* + *k*, C # 0	Jump to address in operand register *d* + *k* if carry ≠ 0
126	P = *dd* + *k*, A = 0	Jump to address in operand register *d* + *k* if A = 0
127	P = *dd* + *k*, A # 0	Jump to address in operand register *d* + *k* if A ≠ 0
130	R = *dd*, C = 0	Return jump to address in operand register *d* if carry = 0
131	R = *dd*, C # 0	Return jump to address in operand register *d* if carry ≠ 0
132	R = *dd*, A = 0	Return jump to address in operand register *d* if A = 0
133	R = *dd*, A # 0	Return jump to address in operand register *d* if A ≠ 0
134	R = *dd* + *k*, C = 0	Return jump to address in operand register *d* + *k* if carry = 0
135	R = *dd* + *k*, C # 0	Return jump to address in operand register *d* + *k* if carry ≠ 0
136	R = *dd* + *k*, A = 0	Return jump to address in operand register *d* + *k* if A = 0
137	R = *dd* + *k*, A # 0	Return jump to address in operand register *d* + *k* if A ≠ 0

I/O Channel Instructions 140 through 177

Instructions 140 through 177 allow I/O Processor (IOP) control of the I/O channel activity. The *d* field in instructions 140 through 157 specifies which I/O channel is addressed. For instructions 160 through 177, the contents of the B register specify the channel. The low-order 4 bits of the function code for the instruction are sent to the channel interface control along with a Go function signal. This 4-bit code is then interpreted by the channel interface control circuits in a manner unique to that channel.

Instructions 140 through 177 can provide the accumulator data to the interface, and the data coming back from the interface can replace the present accumulator contents. Instructions 150 through 153 and 170 through 173 read a 16-bit quantity into the accumulator. This quantity is whatever value the channel interface provides as a result of its interpretation of the channel function. The data transfer is defined in the description for each individual channel. Instructions 140 through 177 can transfer data from the accumulator to the channel interface.

None of the I/O channel instructions involves any significant delay in the execution of the program sequence. There is no mechanism for the I/O channel control to delay execution of further instructions as a result of interpreting the 4-bit code. Delays in executing program functions must be programmed through sampling of the Channel Busy and Done flags or through the equivalent use of the interrupt mechanism.

After issuing any instruction to the I/O channels, allow 1 CP (by issuing a pass instruction, or other instruction) before checking the Channel Busy or Done flags. One CP should also be allowed after any function 6 or function 7 I/O instruction (modifying Channel Interrupt flag) before checking for the channel interrupt number (IOR : 10).

I/O channel instructions 140 through 177 follow.

Instruction IOP	APML	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

Instruction IOP	APML	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
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
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
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
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

INTERFACES 7

To take advantage of the I/O Processor's (IOP's) capabilities, interfaces are required to adapt the IOP to peripheral devices. An interface buffers data, generates control signals for the peripheral device, and multiplexes several devices into the same IOP channel. This section describes the characteristics of the interfaces, gives a table of interface functions, and presents operational information concerning interfaces.

INTERFACE CHARACTERISTICS

Each IOP provides for 40 I/O channels. These channels are addressed by the *d* designator in the program instruction or by the B register contents. Data can be transferred from the IOP accumulator to an interface register or from an interface register to the accumulator. Local Memory ports can be used for block transfers of data into or out of Local Memory. Data transfers and channel interface actions are a function of each interface logic control.

Each interface can interpret up to 16 function signals from the IOP program. These functions are generated by instructions 140 through 177. Interpretation of each function is specifically designated by each interface. However, three functions are common among the interfaces (except the Peripheral Expander) and are described below.

Function	Description
iod : 0 or IOB : 0	Clear the Channel Busy and Done flags and place the channel in an idle status
iod : 6 or IOB : 6	Clear the Channel Interrupt flag for the associated channel, blocking any further interrupt requests from that channel
iod : 7 or IOB : 7	Set the Channel Interrupt Enable flag for the associated channel and enable the interrupt requests from that channel

Each channel interface provides for a Busy flag, normally set during the active period of the channel and cleared during an idle period. The setting and clearing of this flag depends on the channel interface interpretation of the 16 function codes. The Channel Busy flag can be sensed by the IOP program through execution of instructions 041 and 043.

Each channel interface provides for a Done flag, normally used to signal the IOP program when some step of the channel activity has reached a point requiring program action. Setting and clearing of the flag is normally a function of the interface hardware, but the program can set or clear the flag for special purposes. The program senses the state of this flag through instructions 040 and 042. An interrupt is normally generated by the interface hardware when the Channel Done flag and the Channel Interrupt Enable flag are set. The system must have interrupts enabled to be interrupted. When not enabled, however, it can still sense the interrupt waiting through IOR : 10.

INTERFACE FUNCTION CODES

Table 7-1 lists all the currently supported peripheral devices and briefly explains each function code interpretation that is implemented. The APML mnemonic identifies the function. Only the first mnemonic of each type is given. The interface functions for disk storage unit channels are not described below, they are described in Disk Systems Hardware Reference Manual, CRI publication HR-0077.

Table 7-1. Interface Functions

Device	Function Mnemonic	Description
Console keyboard (TIA through TID) (Accumulator channel)	TIA : 0	Clear the Channel Done flag
	TIA : 6	Clear the Channel Interrupt Enable flag
	TIA : 7	Set the Channel Interrupt Enable flag
	TIA : 10	Read data into accumulator and clear Done flag
Console display (TOA through TOD) (Accumulator channel)	TOA : 0	Clear the Channel Busy and Done flags
	TOA : 6	Clear the Channel Interrupt Enable flag
	TOA : 7	Set the Channel Interrupt Enable flag
	TOA : 14	Send accumulator data to display

Table 7-1. Interface Functions (continued)

Device	Function Mnemonic	Description
Expander chassis (EXB) (Accumulator channel)	EXB : 0	Idle the channel
	EXB : 1	Request A input register contents (DIA)
	EXB : 2	Request B input register contents (DIB)
	EXB : 3	Request C input register contents (DIC)
	EXB : 4	Read Busy/done flag, interrupt number
	EXB : 5	Load device address
	EXB : 6	Send interface mask (MSKO)
	EXB : 7	Set interrupt mode
	EXB : 10	Read data bus status
	EXB : 11	Read status 1
	EXB : 13	Read status 2
	EXB : 14	Send data to output A register (DOA)
	EXB : 15	Send data to output B register (DOB)
	EXB : 16	Send data to output C register (DOC)
	EXB : 17	Send control
Input from mainframe I/O channel (CIA through CID) (DMA channel)	CIA : 0	Clear channel
	CIA : 1	Enter Local Memory address, start input
	CIA : 2	Enter parcel count
	CIA : 3	Clear Channel Parity Error flags
	CIA : 4	Clear Ready Waiting flag
	CIA : 6	Clear Interrupt Enable flag
	CIA : 7	Set Interrupt Enable flag
	CIA : 10	Read Local Memory address
	CIA : 11	Read status (ready waiting, parity errors)
Output to mainframe I/O channel (COA through COD) (DMA channel)	COA : 0	Clear channel
	COA : 1	Enter Local Memory address
	COA : 2	Enter parcel count
	COA : 3	Clear Error flag
	COA : 4	Set/clear external control signals
	COA : 6	Clear Interrupt Enable flag
	COA : 7	Set Interrupt Enable flag
	COA : 10	Read Local Memory address
	COA : 11	Read status (4-bit channel data, error)

Table 7-1. Interface Functions (continued)

Device	Function Mnemonic	Description
Input from Central Memory 100 Mbyte channel or SSD (HIA) (DMA channel)	HIA : 0	Clear Channel Busy and Done flags
	HIA : 1	Enter Local Memory address
	HIA : 2	Enter upper Central Memory address
	HIA : 3	Enter lower Central Memory address
	HIA : 4	Read Central Memory; enter block length.
	HIA : 6	Clear Interrupt Enable flag
	HIA : 7	Set Interrupt Enable flag
	HIA : 14	Enter diagnostic mode
Output to Central Memory 100 Mbyte channel or SSD (HOA) (DMA channel)	HOA : 0	Clear Channel Busy and Done flags
	HOA : 1	Enter Local Memory address
	HOA : 2	Enter upper Central Memory address
	HOA : 3	Enter lower Central Memory address
	HOA : 5	Write Central Memory; enter block length.
	HOA : 6	Clear Interrupt Enable flag
	HOA : 7	Set Interrupt Enable flag
	HOA : 14	Enter diagnostic mode
Error logging channel for Serial No. 20 and below (ERA) (Accumulator channel)	ERA : 0	Idle channel
	ERA : 6	Clear Interrupt Enable flag
	ERA : 7	Set Interrupt Enable flag
	ERA : 10	Read error status
	ERA : 11	Read error information (first parameter)
	ERA : 12	Read error information (second parameter)
	ERA : 13	Read error information (third parameter)
Block multiplexer channel (BMA through BMP) (DMA channel)	BMA : 0	Clear channel control
	BMA : 1	Send reset functions
	BMA : 2	Send commands to the control units
	BMA : 3	Read request-in address
	BMA : 4	Clear the Channel Done flag; set the Channel Busy flag for asynchronous I/O.
	BMA : 5	Delay counter diagnostic
	BMA : 6	Clear Channel Interrupt Enable flag
	BMA : 7	Set Channel Interrupt Enable flag
	BMA : 10	Read Local Memory address
	BMA : 11	Read byte count
	BMA : 12	Read status
	BMA : 13	Read input tags
	BMA : 14	Enter Local Memory address
	BMA : 15	Enter byte count
	BMA : 16	Enter device address
	BMA : 17	Enter output tags

CONSOLE KEYBOARD CHANNEL

The IOP provides a number of I/O channels that can be connected to operator console keyboards. Each keyboard is assigned to a separate channel and all can operate independently. Data is transmitted serially from the keyboard to a channel interface register.

The Channel Busy flag is set by the channel hardware at the beginning of the transmission. The Channel Busy flag is cleared and the Channel Done flag is set when the data has been assembled in the interface register.

This channel has a 7-bit interface register which assembles the data for a character associated with a key depression. This data can then be read into the IOP accumulator. Function requests for the first channel are described below.

Function	Description
TIA : 0†	Clear the Channel Done flag
TIA : 6††	Clear the Channel Interrupt Enable flag. The Channel Busy and Channel Done flags are not altered in this process.
TIA : 7††	Set the Channel Interrupt Enable flag. The Channel Busy and Channel Done flags are not altered in this process.
TIA : 10†	Read the contents of the Channel Interface register into the low-order 7-bit positions in the accumulator and clear the high-order bits. This request clears the Channel Done flag.

CONSOLE DISPLAY CHANNEL

The IOP provides a number of I/O channels that can be connected to operator display consoles. Each display is assigned to a separate channel and all can operate independently. Data is transmitted serially from a channel interface register to the display device.

This channel has a 7-bit interface register, which receives data from the IOP accumulator and transmits this data serially to the display device. Function requests for the first channel are described as follows.

---

† Valid Channel Busy and Channel Done flags cannot be read until 1 clock period (CP) after this function is issued.

†† Allow 1 CP before checking the interrupt channel number (IOR : 10).

Function	Description
TOA : 0†	Clear the Channel Busy and Channel Done flags
TOA : 6††	Clear the Channel Interrupt Enable flag. The Channel Busy and Channel Done flags are not altered in this process.
TOA : 7††	Set the Channel Interrupt Enable flag. The Channel Busy and Channel Done flags are not altered in this process.
TOA : 14†	Enter the low-order 7 bits of the accumulator contents in the Channel Interface register. This request sets the Channel Busy flag and clears the Channel Done flag. It begins the transmission of the data from the interface register to the display device, and clears the Channel Busy flag and sets the Channel Done flag when the transmission has been completed.

PERIPHERAL EXPANDER CHANNEL

The Master I/O Processor (MIOP) has a channel connected to a Peripheral Expander. The Peripheral Expander can contain 16 controllers for peripheral devices. Only one controller can be active at one time, but that controller can be servicing more than one peripheral device. Each peripheral unit is assigned a device address and is individually selectable.

The function requests for the Peripheral Expander are listed below.

Function	Description
EXB : 0†††	Idle channel
EXB : 1†††	DIA. Get the data input from the A register of the selected controller.
EXB : 2†††	DIB. Get the data input from the B register of the selected controller.

† Valid Channel Busy and Channel Done flags cannot be read until 1 CP after this function is issued.
†† Allow 1 CP before checking the interrupt channel number (IOR : 10).
††† Allow 1 CP before checking Busy or Done flags.

Function	Description
EXB : 3†	DIC. Get the data input from the C register of the selected controller.
EXB : 4†	Read Busy/Done flags and interrupt number
EXB : 5	Load device address
EXB : 6††	MSKO. Get mask to disable interrupts.
EXB : 7††	Set interrupt mode
EXB : 10	Read data bus status
EXB : 11	Read status 1
EXB : 13	Read status 2
EXB : 14†	DOA. Send data output to the A register of the selected controller.
EXB : 15†	DOB. Send data output to the B register of the selected controller.
EXB : 16†	DOC. Send data output to the C register of the selected controller.
EXB : 17†	Send control

INTERFACE REGISTERS

Each peripheral controller has three interface registers: A, B, and C. These registers are used for control and data communication between the peripheral device and the IOP. The specific uses of the registers are defined by the peripheral device controller.

CHANNEL ASSIGNMENTS

All peripheral devices handled by the Peripheral Expander share the same channel number. Device addresses select among the peripheral units.

---

† Allow 1 CP before checking Busy or Done flags.
†† Allow 1 CP before checking the interrupt channel number (IOR : 10).

EXB : 0 - IDLE CHANNEL

An EXB : 0 function request clears the Peripheral Expander Channel Busy and Channel Done flags, clears the Peripheral Expander Interrupt Enable flag, and clears the interface direct memory access (DMA) Enable flag. The Peripheral Expander channel is then inactive and DMA references through the data channel are not allowed.

EXB : 1 - DIA

An EXB : 1 function requests the contents of the A input register from the selected peripheral controller in the Peripheral Expander. The Peripheral Expander Channel Busy flag sets and the Peripheral Expander Channel Done flag clears, both in the clock period following issue. Since this command requires the use of the data bus, it is delayed by the function delay time (minimum 1 microsecond). When the function completes, the Peripheral Expander Channel Done flag sets and the Peripheral Expander Channel Busy flag clears. At this time, the IOP can follow this instruction with an EXB : 10 function request to load the A input register information into the IOP accumulator.

EXB : 2 - DIB

This function performs exactly as the EXB : 1 function, except that the peripheral controller B input register is sampled.

EXB : 3 - DIC

This function performs exactly as the EXB : 1 function, except that the peripheral controller C input register is sampled.

EXB : 4 - READ BUSY/DONE, INTERRUPT NUMBER

The EXB : 4 function request serves two purposes. First it returns the specified peripheral controller Busy and Done flags to the interface. Second it determines which peripheral controller has the highest priority interrupt, its Done flag set, and its interrupts enabled.

This is a delayed function because it uses the bus data lines from the Peripheral Expander. The Peripheral Expander Channel Busy flag sets and the Peripheral Expander Channel Done flag clears 1 CP after instruction issue. Upon completion, the Peripheral Expander Channel Busy flag is

cleared and the Peripheral Expander Channel Done flag is set. At this time, an EXB : 11 function can be issued to load the peripheral controller Busy flags, Done flags, and the present interrupt device address.

EXB : 5 - LOAD DEVICE ADDRESS

An EXB : 5 function request loads bits 2^0 through 2^5 of the IOP accumulator into the interface device address register. The interface device address register holds the device address of a peripheral controller to which a delayed function is being sent. A delayed function is any function requiring the bus data lines in the Peripheral Expander, thus requiring time to complete. The following are delayed functions: EXB : 1, 2, 3, 4, 6, 14, 15, 16, and 17. The address in the interface device address register must not be changed at any time during the execution of a delayed function. This function does not change the Peripheral Expander Channel Done and the Peripheral Expander Channel Busy flags. The device address is loaded into the interface register in the clock period following instruction issue.

EXB : 6 - MSKO MASK OUT

An EXB : 6 function request sends the present contents of the IOP accumulator to the Peripheral Expander where the 16-bit word acts as a mask to disable interrupts from specific peripheral controllers. Every device controller in the Peripheral Expander has a mask bit. The specific mask bits for peripheral devices and their corresponding octal device codes are given in table 7-2. Since the mask bits are subject to change, the latest documentation for the peripheral interface should be consulted. If the mask bit is set, interrupts from the corresponding peripheral controller are disabled.

The Peripheral Expander Channel Busy flag is set and the Peripheral Expander Channel Done flag is cleared upon initiation of this instruction. Because the mask must be sent throughout the Peripheral Expander through the bus data lines, this instruction is a delayed instruction. When it does complete, the Peripheral Expander Channel Busy flag clears and the Peripheral Expander Channel Done flag sets.

Issuing this instruction clears the Interrupt flag. If an interrupt condition is still present 400 ns after completion, the Interrupt flag sets again.

Table 7-2. Peripheral Device Mask Bits for Interrupt Disabling

Device Code	Mask Bit	Device
60	2^6	AMPEX disk drive
22	2^5	Data General tape drive
17	2^3	Gould printer
70	2^1	Chronolog clock
27	2^8	CDC cartridge disk drive
22	2^5	Kennedy tape drive
15	2^3	Versatec printer
50	2^1	Hayes clock

EXB : 7 - SET INTERRUPT MODE

An EXB : 7 function request enables or disables interrupts for the Peripheral Expander I/O channel and for the peripheral controllers. The accumulator value present when this instruction issues determines the types of interrupts honored. If a bit is set, the interrupts are enabled as follows:

Bit	Meaning
2^0	Enables interrupts from the I/O channel to the IOP
2^1	Enables interrupts from the I/O controllers to the IOP

The I/O Channel Busy and Channel Done flags are not affected by this instruction.

EXB : 10 - READ DATA BUS STATUS

An EXB : 10 function request reads the data on the data bus into the IOP accumulator. This function request is normally used to bring back the data received from the bus by the EXB : 1, 2, or 3 input function requests. The Peripheral Expander I/O Channel Busy and Channel Done flags are not affected by this instruction. The data remains valid until another EXB : 1, 2, or 3 function is issued.

EXB : 11 - READ STATUS 1

An EXB : 11 function request returns the status of the Peripheral Expander I/O channel and the status of the peripheral controller to the IOP. The Channel Busy and Channel Done flags are not affected by this instruction. The status bits are assigned as shown in table 7-3.

EXB : 13 - READ STATUS 2

An EXB : 13 function request returns the Peripheral Expander I/O channel status and the peripheral controller status to the IOP. The Channel Busy and Channel Done flags are not affected by this instruction. The instruction is similar to EXB : 11 except that the contents of the device address register in the interface are returned instead of the interrupt device code number. Status bit assignments are shown in table 7-8.

EXB : 14 - DOA (DATA OUT A)

An EXB : 14 function request sends the contents of the IOP accumulator to the selected peripheral controller in the Peripheral Expander. The data goes into the A register of the peripheral controller whose device address is currently in the device address register.

This function request causes the Peripheral Expander I/O Channel Busy flag to set and the Peripheral Expander I/O Channel Done flag to clear. An EXB : 14 function request requires the data bus in the Peripheral Expander and is a delayed function. When complete, the Peripheral Expander I/O Channel Done flag sets and the Peripheral Expander I/O Channel Busy flag clears.

EXB : 15 - DOB (DATA OUT B)

An EXB : 15 function request sends the contents of the IOP accumulator to the selected peripheral controller in the Peripheral Expander. Data goes into the B register of the addressed device controller. The function request is otherwise the same as the EXB : 14 DOA function request.

EXB : 16 - DOC (DATA OUT C)

An EXB : 16 function request sends the IOP accumulator contents to the C register of the selected peripheral controller in the Peripheral Expander. Otherwise, the function request is the same as the EXB : 14 DOA function request.

Table 7-3. Read Status 1 Bit Assignments

Bit	Function
2^0	Interrupting device code bit 2^0
2^1	Interrupting device code bit 2^1
2^2	Interrupting device code bit 2^2
2^3	Interrupting device code bit 2^3
2^4	Interrupting device code bit 2^4
2^5	Interrupting device code bit 2^5
2^6	Unassigned
2^7	DMA enabled
2^8	Expander I/O channel interrupts enabled
2^9	Controller interrupts enabled
2^{10}	Function active - delayed function executing
2^{11}	Expander I/O Channel Busy flag
2^{12}	Expander I/O Channel Done flag
2^{13}	INTR - Interrupt request from peripheral interface
2^{14}	SELB - Select Busy flag of addressed device
2^{15}	SELD - Select Done flag of addressed device

Table 7-4. Read Status 2 Bit Assignments

Bit	Function
2^0	Device address bit 2^0
2^1	Device address bit 2^1
2^2	Device address bit 2^2
2^3	Device address bit 2^3
2^4	Device address bit 2^4
2^5	Device address bit 2^5
2^6	Unassigned
2^7	DMA enabled
2^8	Expander I/O channel interrupts enabled
2^9	Controller interrupts enabled
2^{10}	Function active - delayed function executing
2^{11}	Expander I/O Channel Busy flag
2^{12}	Expander I/O Channel Done flag
2^{13}	INTR - Interrupt request from peripheral interface
2^{14}	SELB - Select Busy flag of addressed device
2^{15}	SELD - Select Done flag of addressed device

EXB : 17 - SEND CONTROL

The peripheral controllers use four control signals in the Peripheral Expander: I/O reset, Pulse, Clear, and Start. Each controller can have different uses for these signals. This function request sends the control signal selected by the bits in the accumulator at instruction issue. Control signals are delayed instructions. The I/O reset signal clears all controllers. The Peripheral Expander I/O Channel Busy flag sets at issue time and the Peripheral Expander I/O Channel Done flag clears. Then when the command has completed, the Channel Done flag sets and the Channel Busy flag clears. The accumulator bits set control signals as shown in table 7-5.

Table 7-5. Accumulator Bit Control Signals

Bit	Function
2^0	Start (S)
2^1	Clear (C)
2^2	Pulse (P)
2^3	I/O reset (IORST)

DELAYED FUNCTIONS

Several functions require the data bus in the Peripheral Expander. These functions need extra time to complete and may have to wait for the data bus to be free. The completion of a delayed function is shown by setting the Peripheral Expander I/O Channel Done flag. The Channel Done flag for the preceding delayed function must be received at the IOP before sending another delayed function. The delayed EXB functions are: 1, 2, 3, 4, 6, 14, 15, 16, and 17.

TRANSFER SPEEDS

The Peripheral Expander interface can sustain a transfer rate of 16 Mbits/s from the IOP to the expander chassis. In the reverse direction, it can sustain a speed of approximately 14.5 Mbits/s. This speed is achieved by using the data channel mode which transfers a 16-bit parcel every microsecond. The programmed I/O mode can also reach these speeds.

CHANNEL FOR INPUT FROM CRAY MAINFRAME CHANNEL

The MIOP can have one or more channels dedicated to receiving data from a Cray mainframe I/O channel. Data is transferred in block mode directly into IOP Local Memory. The functions are listed as follows.

Function	Description
CIA : 0†	Clear channel
CIA : 1†	Enter Local Memory address, start transfer
CIA : 2	Enter parcel count
CIA : 3	Clear Channel Parity Error flags
CIA : 4	Clear Ready Waiting flag
CIA : 6††	Clear Interrupt Enable flag
CIA : 7††	Set Interrupt Enable flag
CIA : 10	Read Local Memory address
CIA : 11	Read Ready Waiting/Error flags

LOCAL MEMORY ADDRESS REGISTER

The Local Memory Address register contains the Local Memory address for the next Local Memory reference. It is loaded with the Local Memory starting address at the initiation of the input transfer. Because memory references are done in bursts of 4 parcels, the low-order 2 bits of the starting address are forced to 0 when loaded into the register. Upon completion of a reference, the register address is increased by 4. If the input transfer stops on some boundary other than four, the final memory reference stores undefined parcels. The number of defined parcels can be determined by reading the final memory address, which is equal to the address of the last defined parcel, plus 1.

CIA : 0 - CLEAR CHANNEL

A CIA : 0 function request clears the interface Channel Busy and Channel Done flags and aborts any transfer in progress. The new states of busy and done are not valid until the second CP following this function. One CP must occur after issuing this function before sampling these flags.

† Allow 1 CP before checking Busy or Done flags.
†† Allow 1 CP before checking the interrupt channel number (IOR : 10).

CIA : 1 - ENTER LOCAL MEMORY ADDRESS

A CIA : 1 function request enters the accumulator contents into the 16-bit Local Memory address register (forcing the 2 low-order bits to 0) and starts an input transfer from a Cray mainframe. The Busy flag is set and the Done flag is cleared. Upon receipt of 4 parcels, a memory reference is made. Then another 4 parcels are received and stored. This cycle continues until the parcel count is 0 or until a disconnect is received from a Cray mainframe I/O channel. At that time, the Done flag is set and the Busy flag is cleared.

CIA : 2 - ENTER PARCEL COUNT

A CIA : 2 function request stores the accumulator contents into the interface Parcel Count register. This value is a positive count of the number of parcels to be transferred. The status of the Busy and Done flags is not affected.

CIA : 3 - CLEAR CHANNEL PARITY ERROR FLAGS

Four parity bits protect the 16-bit parcels on a Cray mainframe channel. A parity error in a 4-bit group causes one Parity Error flag to set. This function request clears all four Parity Error flags. A parity error does not affect the status of the Channel Busy and Channel Done flags. Similarly, issuing this function request does not affect the states of the Channel Busy and Channel Done flags.

CIA : 4 - CLEAR READY WAITING

If the IOP input channel is inactive and a Ready signal is received from a Cray mainframe, the channel sets a Ready Waiting flag. To resynchronize the transfer, this Ready signal must be discarded. The CIA : 4 function request clears the Ready Waiting flag. If the Ready signal is not discarded and the input channel is started, the data on the lines is sampled as the first parcel of the transfer. The Channel Busy and Channel Done flags are not affected by this function.

CIA : 6 - CLEAR INTERRUPT ENABLE FLAG

A CIA : 6 function request disables the interface from interrupting the IOP. The channel can still be monitored through the Busy and Done flag status. The Channel Busy and Channel Done flags are not affected by this function.

CIA : 7 - SET INTERRUPT ENABLE FLAG

A CIA : 7 function request enables interrupts on the interface. The channel interrupts whenever the Channel Done flag sets. The Channel Busy and Channel Done flags are not affected by this function.

CIA : 10 - READ MEMORY ADDRESS

A CIA : 10 function request transfers the contents of the interface Local Memory address register into the accumulator. The address is one greater than the address of the last parcel stored. Since a memory reference is four parcels of data, there are undefined parcels of data written into memory if the transfer length is not a multiple of 4. The Channel Busy and Channel Done flags are not affected by this function.

CIA : 11 - READ READY WAITING/ERROR FLAGS

A CIA : 11 function request reads the contents of the interface status register into the accumulator. The status bit assignment is listed in table 7-6.

Table 7-6. Ready Waiting/Error Flags

Bit	Meaning
2^0	Channel Parity Error flag for bits 2^0-2^3
2^1	Channel Parity Error flag for bits 2^4-2^7
2^2	Channel Parity Error flag for bits 2^8-2^{11}
2^3	Channel Parity Error flag for bits 2^{12}-2^{15}
2^{15}	Ready waiting flag

CHANNEL FOR OUTPUT TO CRAY MAINFRAME CHANNEL

The IOP can have one or more channels for sending data to a Cray mainframe input channel. Data is transferred in block mode directly from Local Memory. The functions are listed as follows.

Function	Description
COA : 0†	Clear channel
COA : 1†	Enter Local Memory address, start transfer
COA : 2	Enter parcel count
COA : 3	Clear Error flag
COA : 4	Set/clear external control signals
COA : 6††	Clear Interrupt Enable flag
COA : 7††	Set Interrupt Enable flag
COA : 10	Read Local Memory address
COA : 11	Read error flags

LOCAL MEMORY ADDRESS REGISTER

The Local Memory Address register contains the Local Memory address for the next Local Memory reference. The register is loaded with the Local Memory starting address at the initiation of the output transfer. The 2 low-order bits of the starting address are forced to 0 when entered in the register because the memory references are in bursts of 4 parcels and, upon completion of a memory reference, the register address is increased by 4. Upon completion of the output, the Local Memory address in the register is one greater than the address of the first parcel sent.

COA : 0 - CLEAR CHANNEL

A COA : 0 function request clears the Channel Busy and Channel Done flags and aborts any transfer. The Busy and Done flags are not valid for sampling until the second clock period following completion of this function.

† Allow 1 CP before checking Busy and Done flags.
†† Allow 1 CP before checking the interrupt channel number (IOR : 10).

COA : 1 - ENTER LOCAL MEMORY ADDRESS

A COA : 1 function request loads the current contents of the accumulator into the 16-bit Local Memory Address register (forcing the 2 low-order bits to 0) and starts the transfer to the Cray mainframe. The Busy flag sets and the Done flag clears. The interface makes a memory reference starting at the address contained in the Local Memory Address register and outputs 4 parcels of data. Then another reference reads out another burst of 4 parcels. When the parcel count stored in the interface register has been reached, the transfer stops. The Done flag sets and the Busy flag clears.

COA : 2 - ENTER PARCEL COUNT

A COA : 2 function request enters the accumulator contents into the interface Parcel Count register. This value is a positive count of the number of parcels to be transferred. The Channel Busy and Channel Done flags are not affected by this function.

COA : 3 - CLEAR ERROR FLAG

A COA : 3 function request clears the Sequence Error flag. The Sequence Error flag sets when the interface receives a Resume signal from the Cray mainframe and the interface is not busy or a Local Memory reference is in progress. If the interface Interrupt Enable flag is set, the Sequence Error flag causes an interrupt. The sequence error has no affect on the Channel Busy and Channel Done flags. The Channel Busy and Channel Done flags are not affected by this function.

COA : 4 - SET/CLEAR EXTERNAL CONTROL SIGNALS

A COA : 4 function request sends control signals to the Cray mainframe. The signal selection is governed by set bits in the accumulator as shown in table 7-7.

A Hold Disconnect signal stops the automatic disconnect that is sent at the end of a transfer. A Write Disconnect signal sends a disconnect to a Cray mainframe with no data transferred. The Error Channel Resume signal is reserved for maintenance use with configurations without an error logging channel. The RTC Interrupt signal is reserved for use on systems where a programmable clock is not present.

The accumulator signal select bits are held in an interface register until they are altered by another COA : 4 function request. The Channel Busy and Channel Done flags are not affected by this function.

Table 7-7. External Control Signal Bits

Bit	Control Signal
2^8	Write Disconnect
2^9	Hold Disconnect
2^{10}	Unused
2^{11}	Dead Dump
2^{12}	RTC Interrupt
2^{13}	Error Channel Resume
2^{14}	I/O Master Clear
2^{15}	Cray Mainframe Master Clear

COA : 6 - CLEAR INTERRUPT ENABLE FLAG

A COA : 6 function request clears the Channel Interrupt Enable flag to prevent the interface from interrupting the IOP. The channel can be monitored through the Busy and Done flags and through the status information returned by the COA : 11 function.

COA : 7 - SET INTERRUPT ENABLE FLAG

A COA : 7 function request sets the Interrupt Enable flag to allow the interface to interrupt the IOP. With interrupts enabled, the interface interrupts whenever the Done flag sets or whenever a sequence error occurs. This function does not affect the Channel Busy and Channel Done flags.

COA : 10 - READ LOCAL MEMORY ADDRESS

A COA : 10 function request enters the content of the Local Memory Address register into the accumulator. The address entered is one greater than the address of the last parcel transferred from Local Memory. The Channel Busy and Channel Done flags are not affected by this command.

COA : 11 - READ ERROR FLAGS

A COA : 11 function request reads the status of the interface into the accumulator. The significance of each bit is listed in table 7-8.

Table 7-8. Error Flags

Bit	Meaning
2^0	4-bit channel data bit 2^0
2^1	4-bit channel data bit 2^1
2^2	4-bit channel data bit 2^2
2^3	4-bit channel data bit 2^3
2^{15}	Sequence error

The Channel Busy and Channel Done flags are not affected by this command. The 4-bit channel is reserved for maintenance use with configurations without an error logging channel.

MEMORY (100 MBYTE) CHANNEL

One or two Memory (100 Mbyte) Channels are used for transferring data between a Cray mainframe's Central Memory and an IOP's Local Memory or between a Solid-state Storage Device (SSD) and an IOP's Local Memory. A Memory (100 Mbyte) Channel consists of an input channel and an output channel. The input channel carries data from the Central Memory or SSD to Local Memory. The output channel passes data from Local Memory to the Central Memory or SSD. Each channel carries 64 bits of data in parallel with 8 check bits. All transfers take place in 16-word bursts. A separate 12-bit channel is provided for both the input channel and the output channel for carrying address information. A data rate of approximately 100 Mbyte/s is possible on either the input channel or the output channel.

The input and output are more complex than for the normal Cray mainframe channels. They have more control signals and more data protection information. The signals and their timing are described below, followed by an explanation of the IOP instructions that control a Memory (100 Mbyte) Channel.

SIGNAL DESCRIPTIONS

Signals for the input and output channels are described in the following paragraphs. Figure 7-1 depicts the Memory (100 Mbyte) Channel signals and their directions.

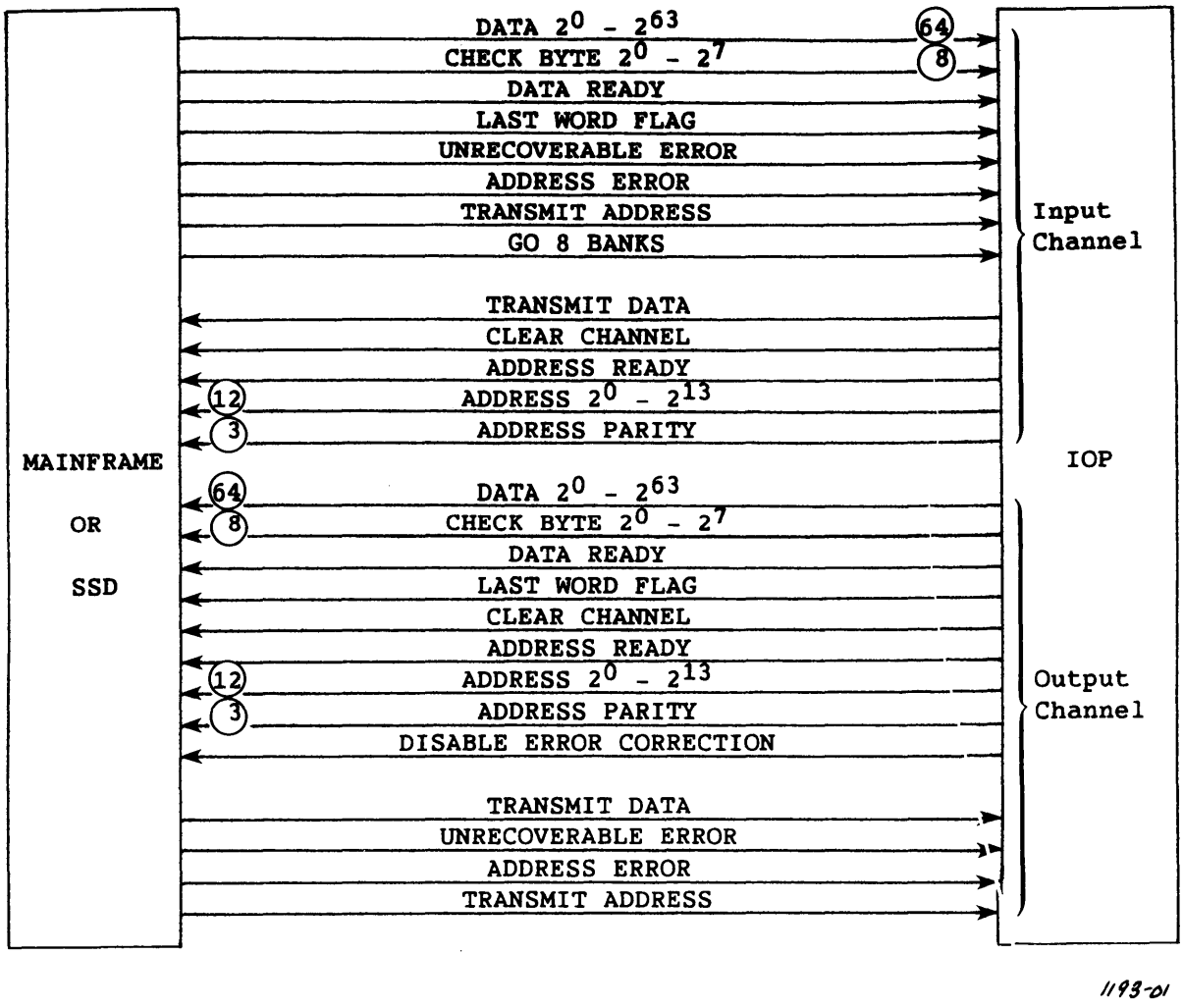


Figure 7-1. Memory (100 Mbyte) Channel Signals

Cray mainframe or SSD to I/O Processor input channel

The input channel signals (from Central Memory or an SSD to IOP) and their timing are described below.

Data (to IOP) - The data is passed in 64-bit words over 64 lines. Data is valid 2 CPs after the leading edge of the Data Ready signal and stays valid for 6 CPs.

Check Byte (to IOP) - The Check Byte is the 8 error check bits read out from Central Memory or an SSD. It is passed to the IOP to verify the accuracy of the received data. The Check Byte has the same timing and valid points as the Data Signal.

Data Ready (to IOP) - This signal indicates data is valid on the cable. The leading edge of the Data Ready signal precedes the Data signals by 2 CPs and is true for 1 CP if the signal is from Central Memory or 2 CPs if the signal is from an SSD. The minimum period for Data Ready signals is 6 CPs.

Last Word (to IOP) - This signal indicates the last data word of the complete transfer is on the Data lines. This also means the Central Memory or SSD word count is 0. This signal has the same timing and duration as the Data signals.

Unrecoverable Error (to IOP) - This signal indicates an unrecoverable error occurred in the transfer at the Central Memory or SSD. The channel goes inactive and remains so until a Clear Channel signal is sent to Central Memory or SSD from the IOP. Unrecoverable error conditions include:

- Address parity error
- Transmit Data signal received at Central Memory or SSD and less than three Address Ready signals were received
- More than three Address Ready signals received at Central Memory or SSD
- Uncorrectable data error from Central Memory or SSD

The Unrecoverable Error signal continues until a Clear Channel signal is sent.

Address Error (to IOP) - The Address Error signal indicates the Unrecoverable Error was an address error, from conditions 1, 2, or 3 of the error conditions listed previously in Unrecoverable Error. The Address Error signal is sampled when the leading edge of the Unrecoverable Error signal appears.

Transmit Address (to IOP) - This signal indicates the Central Memory or SSD side is inactive and that a transfer can be initiated by sending an address on the Address lines. A Transmit Address signal clears after three Address Ready signals and does not set again until after the last data word has been transferred to the IOP. This signal also clears if there is an error condition on the Central Memory or SSD side of the channel and does not set again until a Clear Channel signal is sent.

Go 8 Banks (to IOP) - When connected to Central Memory, this signal indicates operation in the 8-bank mode. All transmitted block sizes are reduced to 8 words. When connected to an SSD, this signal is always a logical 0.

Transmit Data (to Central Memory or SSD) - This signal indicates the IOP can accept a block of data from the Central Memory or SSD. The Transmit Data signal is clear when the channel is inactive and does not set for at least 8 CPs after the third Address Ready pulse trailing edge. The signal remains set until the first Data Ready signal is received for that data block. Until the end of the transmission, Transmit Data is asserted whenever the IOP channel's input data buffer can receive 16 words (8 words if in 8-bank mode) without overflowing. Transmit Data is sampled at Central Memory or SSD at the beginning of each data block. If the Transmit Data signal is set, the 16-word (or 8-word) data block is transmitted. If this signal is cleared at that time, no data is transmitted until the Transmit Data signal sets.

NOTE FOR CRAY-1 S AND SSD

The number of 64-bit data words sent in the first block equals 16 minus the number specified by bits 2^0 and 2^1 of the Central Memory starting address or bits 2^6 and 2^7 of the SSD starting address. The size of the first block is manipulated in this manner to adjust the blocks to 16-bank Central Memory boundaries. The next block begins with section 0. If the Go 8 Banks signal is set, the number of data words in the first block is 8 minus the address 2^0 value. The SSD maintains this Go 8 Banks signal at a zero level.

Clear Channel (to Central Memory or SSD) - This signal clears all error conditions in the channel. It remains set a minimum of 8 CPs. An Address Ready signal must lag the trailing edge of the Clear Channel signal by at least 8 CPs.

Address Ready (to Central Memory or SSD) - This signal indicates Address lines will soon have valid address data. The signals leading edge leads the address information by 2 CPs and stays set for 1 CP. The minimum period for Address Ready signals is 6 CPs, leading edge to leading edge.

Three Address Ready signals initiate a transfer. The first signal accompanies bits 2^{10} - 2^{21}† of the beginning Central Memory address, or bits 2^{16} - 2^{27}† of the beginning SSD address, on the Address lines.

The second signal to Central Memory accompanies bits 2^0 - 2^9 of the beginning Central Memory address, and bits 2^{12} - 2^{13} of the transfer word count. The second signal to an SSD accompanies bits 2^6 - 2^{15} of the beginning SSD address and bits 2^{12} - 2^{13} of the transfer word count.

† If the IOS has the address expansion option, two additional bits are transferred.

The third signal accompanies bits 2^0 - 2^{11} of the transfer word count. For an SSD, bits 2^0 - 2^5 of the transfer word count must always be zeros. See figures 7-2 and 7-3.

Address (to Central Memory or SSD) - The 12 Address lines carry Central Memory or SSD starting address and the transfer word count. Information carried on the Address lines for each of the three address transfers is shown in figures 7-2 and 7-3. It is valid for 6 CPs. The Address signals lag the leading edge of the Address Ready signal by 2 CPs.

Address Parity (to Central Memory or SSD) - These signals provide odd parity for each group of 4 address bits as follows:

Parity Bit	Information Bits
2^0	2^0 through 2^3
2^1	2^4 through 2^7
2^2	2^8 through 2^{11}
2^3	2^{12} through 2^{13}

The Address Parity signals have the same timing as Address signals. They are valid for 6 CPs.

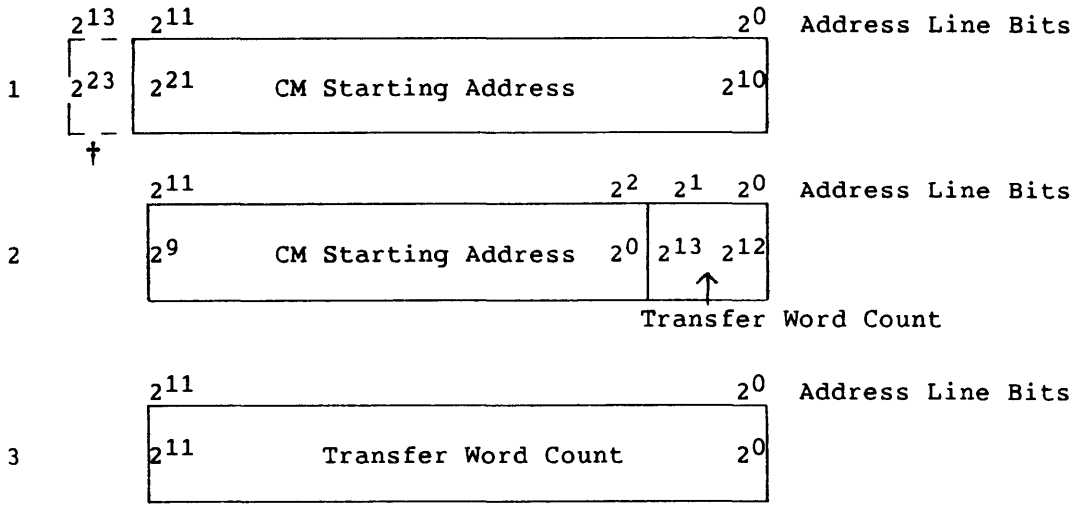


Figure 7-2. Address and Word Count Formats for Central Memory

† Some I/O Subsystems have an address expansion option that transfers additional address information with each address ready. This option is used only with large memory mainframes and SSDs.

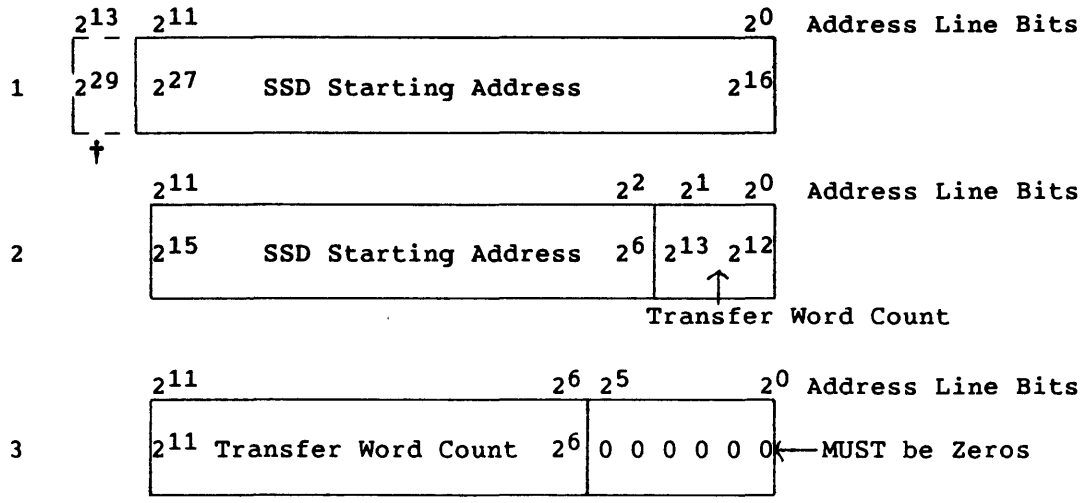


Figure 7-3. Address and Word Count Formats for SSD

I/O Processor Output Channel to Cray mainframe or SSD

The Output Channel signals (from the IOP to Central Memory or SSD) and their timings are described below.

Data (to Central Memory or SSD) - Data passes in 64-bit words over 64 lines. Data is valid 2 CPs after the Data Ready signal leading edge and data stays valid for 6 CPs.

Check Byte (to Central Memory or SSD) - The Check Byte is the 8 error check bits generated by the output side of the channel. The error code generation methods used are the same on each side of the channel. The Check Byte is passed to the Central Memory or SSD with the same timing and valid points as the Data signals.

Data Ready (to Central Memory or SSD) - This signal indicates data on the cable is valid. The leading edge of the Data Ready signal precedes the Data signals by 2 CPs and the Data Ready signal is set for 1 CP. The minimum period for Data Ready signals is 6 CPs.

Last Word Flag (to Central Memory or SSD) - This signal indicates the last data word of the complete transfer is on the Data lines. This also means the IOP word count is 0. The Last Word Flag signal has the same timing and duration as the Data signals.

† Some I/O Subsystems have an address expansion option that transfers additional address information with each address ready. This option is used only with large memory mainframes and SSDs.

Clear Channel (to Central Memory or SSD) - This signal clears all error conditions in the channel. It remains set a minimum of 8 CPs. An Address Ready signal must lag the trailing edge of the Clear Channel signal by at least 8 CPs.

Address Ready (to Central Memory or SSD) - This signal indicates information will soon be valid on the Address lines. The leading edge of the Address Ready signal leads the valid address information by 2 CPs and is set for 1 CP. The minimum period for sequential Address Ready signals is 6 CPs.

Three Address Ready signals initiate a data block transfer. The first Address Ready pulse is interlocked to the Transmit Address signal and is sent only when the Transmit Address signal is set. Once the first pulse is sent, the next two are unconditionally sent. The first Address Ready signal to Central Memory precedes bits 2^{10} through 2^{21} of the Central Memory starting address. The first Address Ready signal to an SSD precedes bits 2^{16} through 2^{27} of the SSD starting address.

The second Address Ready signal to Central Memory precedes bits 2^{0} through 2^{9} of the Central Memory starting address, and bits 2^{12} through 2^{13} of the transfer word count. The second Address Ready signal to an SSD precedes bits 2^{6} through 2^{15} of the SSD starting address, and bits 2^{12} through 2^{13} of the transfer word count.

The third Address Ready signal to Central Memory precedes bits 2^{0} through 2^{11} of the transfer word count. The third Address Ready signal to an SSD also precedes bits 2^{0} through 2^{11} of the transfer word count.

However, bits 2^{0} through 2^{5} of the SSD transfer word count must always be zeros. If bits 2^{0} through 2^{5} are nonzero, an SSD transfer length error results.

See figures 7-2 and 7-3 for the address information formats that follow each Address Ready signal.

Address (to Central Memory or SSD) - The 12 Address lines carry the Central Memory or SSD starting address and the transfer word count. The Address signals lag the leading edge of the Address Ready signal by 2 CPs and then stay set for 6 CPs. The formats of the information carried in each of three address information transfers are shown in figures 7-2 and 7-3.

Address Parity (to Central Memory or SSD) - These signals provide odd parity for each group of four Address signals as follows:

Parity Bit	Information Bits
2^0	2^0 through 2^3
2^1	2^4 through 2^7
2^2	2^8 through 2^{11}
2^3	2^{12} through 2^{13}

The Address Parity signals have the same timing as Address Signals. They are valid for 6 CPs.

Disable Error Correction (to Central Memory or SSD) - This signal is used for diagnostic purposes only. When connected to Central Memory it disables the single-error correction/double-error detection (SECDED) circuitry used on the data channel. When connected to an SSD, it only prevents double-bit channel errors from generating an Unrecoverable Error. The SSD Memory still invokes SECDED and data errors are still reported to the SSD error logger.

Transmit Data (to IOP) - This signal indicates the Central Memory or SSD can accept a block of data from the IOP. This signal remains set as long as the Central Memory or the SSD is capable of receiving the next 16-word burst (8 words in 8-bank mode) without overflowing its input data buffers. Transmit data is sampled at the beginning of each data block. If the Transmit Data signal is set and the Central Memory is using 16 banks, a 16-word data block is transmitted. An 8-word data block is transmitted if Central Memory is using 8 banks. The Transmit Data signal to an SSD causes a 16-word data block to be transferred. If the Transmit Data signal is cleared when sampled, no data transfers until the Transmit Data signal sets.

NOTE FOR CRAY-1 S AND SSD

The number of 64-bit words sent in the first block of a Central Memory transfer equals the number of banks (16 or 8), minus the number specified by bits 2^0 and 2^1 (16 banks) or by bit 2^0 (8 banks) of the Central Memory starting address. This procedure adjusts the blocks to 16-bank or 8-bank boundaries so that the next block begins with bank 0.

In the case of an SSD transfer, the number of 64-bit words is 16 minus the number specified by bits 2^6 and 2^7 of the SSD starting address.

Unrecoverable Error (to IOP) - This signal indicates an unrecoverable error occurred in the transfer at the Central Memory or SSD side. The channel goes inactive and remains so until a Clear Channel signal is sent to the Central Memory or the SSD.

Unrecoverable error conditions include:

- Address parity error
- Data Ready received and less than three Address Ready signals
- More than three Address Ready signals received
- Uncorrectable data error on channel
- Transfer word count = 0 and no Last Word Flag signal
- Transfer word count ≠ 0 and Last Word Flag signal

The Unrecoverable Error signal remains set until a Clear Channel signal is sent.

Address Error (to IOP) - The Address Error signal indicates the Unrecoverable Error was an address error from conditions 1, 2, or 3 of the Unrecoverable Error list shown above. The Address Error signal is sampled when the leading edge of the Unrecoverable Error signal appears.

Transmit Address (to Central Memory or SSD) - This signal indicates the Central Memory or the SSD side is inactive and a transfer can be initiated by sending data over the Address lines. This signal clears after three Address Ready signals and does not set again until the last word of the complete channel transfer has been stored in Central Memory or in an SSD. If an error condition is discovered on the Central Memory or SSD side, the Transmit Address signal clears and does not set again until the Clear Channel signal is sent.

MEMORY (100 MBYTE) CHANNEL FUNCTIONS FOR INPUT FROM CENTRAL MEMORY OR SSD

The input functions for the Memory (100 Mbyte) Channel are summarized below and described on the following pages.

NOTE

For a transfer from SSD, the transfer starts on a 64-word SSD boundary and must be a multiple of 64 words.

Function	Description
HIA : 0†	Clear Channel Busy and Channel Done flags
HIA : 1	Enter Local Memory address
HIA : 2	Enter upper Central Memory or SSD address
HIA : 3	Enter lower Central Memory or SSD address
HIA : 4†	Read Central Memory or SSD, enter block length
HIA : 6††	Clear Interrupt Enable flag
HIA : 7††	Set Interrupt Enable flag
HIA : 14	Enter diagnostic mode

INTERFACE REGISTERS

Three interface registers control transfers over the Memory (100 Mbyte) Channel. The Central Memory or SSD starting address is held in a 22-bit register. Two separate functions, HIA : 2 and HIA : 3, each load portions of the address information. The IOP Local Memory starting address is held in a 16-bit register that is loaded by an HIA : 1 function. The third register holds 14 bits of the transfer word count or block length entered by the HIA : 4 function.

† Allow 1 CP before checking Busy or Done flags.
†† Allow 1 CP before checking the interrupt channel number (IOR : 10)

HIA : 0 - CLEAR CHANNEL BUSY AND CHANNEL DONE FLAGS

This function clears the Channel Busy and Channel Done flags. It causes an error condition if issued while this channel is active. HIA : 0 must be used to clear an error condition on the channel.

HIA : 1 - ENTER LOCAL MEMORY STARTING ADDRESS

This function enters the contents of the IOP accumulator into the Local Memory starting address register. Bits 2^0 and 2^1 of the address are forced to 0. This command does not alter the state of the Channel Busy or Channel Done flags. If the HIA : 1 function is given while this channel is active, an error condition occurs.

HIA : 2 - ENTER UPPER CENTRAL MEMORY OR SSD ADDRESS

This function enters bits 2^0 through 2^{14} of the IOP accumulator into the 2^9 through 2^{23} positions of the Central Memory starting address register. For an SSD, this function enters bits 2^0 through 2^{14} of the IOP accumulator into the 2^{15} through 2^{29} bit positions of the SSD starting address register. The states of the Channel Busy and Channel Done flags are not altered. The HIA : 2 function causes an error condition if issued while this channel is active.

****

CAUTION: Unused bits must be zeroed.

****

HIA : 3 - ENTER LOWER CENTRAL MEMORY OR SSD ADDRESS

This function enters bits 2^0 through 2^8 of the IOP accumulator into bits 2^0 through 2^8 of the Central Memory starting address register. For an SSD, bits 2^0 through 2^8 of the IOP accumulator enter bit positions 2^6 through 2^{14} of the SSD starting register. The states of the Channel Busy and Channel Done flags are not altered. This function causes an error condition if issued when the input channel is active.

****

CAUTION: Unused bits must be zeroed.

****

NOTE

The 22-bit SSD address entered through the HIA : 2 and HIA : 3 functions is the SSD starting address divided by 100_8.

HIA : 4 - READ CENTRAL MEMORY OR SSD, ENTER BLOCK LENGTH

This function enters the low-order bits (2^0 through 2^{13}) of the IOP accumulator into the Block Length register. (This value is treated as a count of 64-bit words with a zero value indicating the maximum 16,384 words.) The Busy flag is set, the Done flag is cleared, and the transfer is initiated. Upon completion, the Done flag sets and the Busy flag clears. If during the transfer an unrecoverable error occurs, the transfer terminates, the Done flag sets, and the Busy flag remains set. This function causes an error condition if issued while the input channel is active.

NOTE

When connected to an SSD, bits 2^0 through 2^5 of the word count must be zeros to avoid a block length error.

HIA : 6 - CLEAR INTERRUPT ENABLE

This function clears the Interrupt Enable flag for the channel. The states of the Busy and Done flags are not altered.

HIA : 7 - SET INTERRUPT ENABLE

This function sets the Interrupt Enable flag for the channel. The states of the Busy and Done flags are not altered.

HIA : 14 - ENTER DIAGNOSTIC MODE

This function enters bits 2^0 through 2^2 of the IOP accumulator into the diagnostic mode register. The modes are summarized in table 7-9.

Only one mode is valid at a time. All diagnostic modes are cleared with a Master Clear, or alternately, by doing HIA : 0, followed by HIA : 14 with the accumulator = 0. Mode 0 is active until any other mode is selected. This function is for maintenance purposes only.

Table 7-9. Input Channel Diagnostic Modes

Mode Designator	Function
0	Set Last Word Flag signal (only if channel is active)
1	Disable Last Word Flag signal from Central Memory or SSD
2	Force constant Address Ready signal
3	Disable third Address Ready signal
4	Transfer first address with zero parity bits, third address with one parity bits, and disable Transmit Data signal to Central Memory or SSD
5	Force Clear Channel signal without inactivating IOP
6	Disable Block length = 0
7	Disable IOP data error correction and detection

MEMORY (100 MBYTE) CHANNEL INPUT ERROR PROCESSING

If an unrecoverable error occurs in an input transfer from the Central Memory or SSD, the Busy and Done flags are both set. An error code is generated and sent to an error log. The error codes are summarized in table 7-10.

MEMORY (100 MBYTE) CHANNEL INPUT SEQUENCE

Figure 7-4 shows the sequence of signals for transferring data from the Central Memory or an SSD to Local Memory.

The only recoverable error is a corrected single-bit data error. This type of error is ignored by the channel because the data is corrected automatically; the syndrome information is sent to an error log.

Table 7-10. Input Channel Error Codes

Error Code	Name	Condition
0	Single-bit Error	A single-bit memory error was discovered and corrected.
1	Function Error	A function 0, 1, 2, 3, or 4 was issued while the channel was active.
2	Active Error	Central Memory or SSD side went inactive while the IOP side was still active.
3	Transmit Address Time-out	An IOP input was initiated but Central Memory or SSD did not send a Transmit Address signal within 2 ms.
4	Central Memory or SSD Address Error	Central Memory or SSD side received greater or less than three Address Ready signals, or a parity error occurred in one of the address channel transfers.
5	Central Memory or SSD Data Error	Central Memory or SSD side has multiple data error from memory or the IOP side received data with a multiple error.
6	Block Length Error	The IOP received the last word and the block length count was not 0 or the last word was not received and the block length count was equal to 0.
7	Data Ready Time-out	The IOP did not receive data within 2 ms.

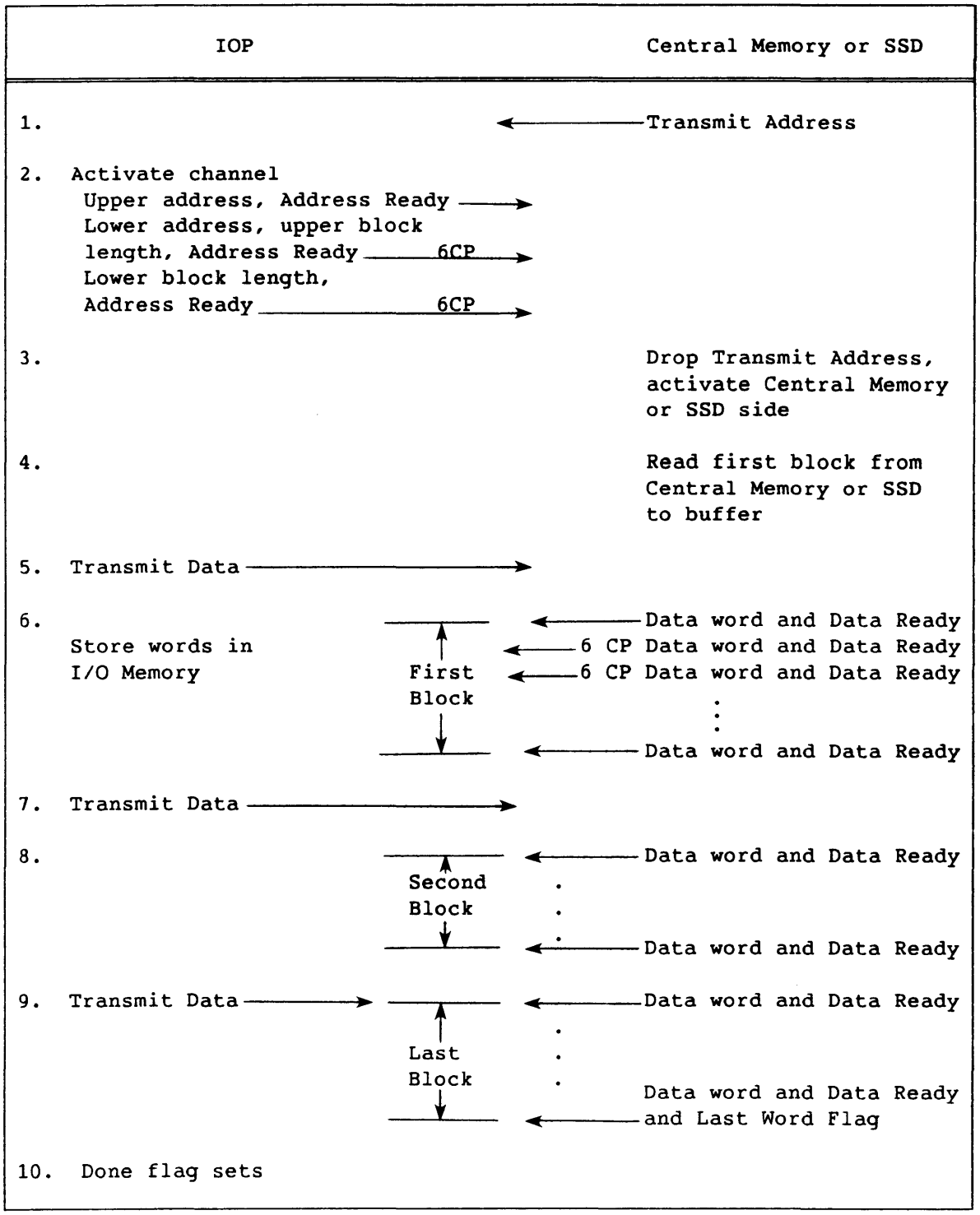


Figure 7-4. Memory (100 Mbyte) Channel Sequence on Input to I/O Processor

MEMORY (100 MBYTE) CHANNEL FUNCTIONS FOR OUTPUT TO CENTRAL MEMORY OR SSD

The output functions for the Memory (100 Mbyte) Channel are summarized below and described on the following pages.

NOTE

For a transfer to SSD, the transfer starts on a 64-word SSD boundary and must be a multiple of 64 words.

Function	Description
HOA : 0†	Clear Channel Busy and Channel Done flags
HOA : 1	Enter Local Memory address
HOA : 2	Enter upper Central Memory or SSD address
HOA : 3	Enter lower Central Memory or SSD address
HOA : 5†	Write Central Memory or SSD, enter block length
HOA : 6††	Clear Interrupt Enable flag
HOA : 7††	Set Interrupt Enable flag
HOA : 14	Enter diagnostic mode

INTERFACE REGISTERS

The interface has three registers to control the output transfers. One 22-bit register holds the Central Memory or SSD starting address. Two separate functions, HOA : 2 and HOA : 3, each load portions of address information. The IOP Local Memory starting address is held in a 16-bit register which is entered by the HOA : 1 function. A 14-bit register holds the transfer word count, or block length in 64-bit words, that is loaded by the HOA : 5 command.

---

† Allow 1 CP before checking Busy or Done flags.
†† Allow 1 CP before checking the interrupt channel number (IOR : 10).

HOA : 0 - CLEAR CHANNEL BUSY, DONE FLAGS

This function idles the channel by clearing the Busy and Done flags. This function causes an error condition if issued while the output channel is active. This function must be used to clear an error condition in the channel.

HOA : 1 - ENTER LOCAL MEMORY ADDRESS

This function enters the value of the IOP accumulator into the Local Memory address register. Bits 2^0 through 2^1 are forced to a zero value. The states of the Busy and Done flags are not altered. This function causes an error condition if issued when the output channel is active.

HOA : 2 - ENTER UPPER CENTRAL MEMORY OR SSD ADDRESS

This function enters bits 2^0 through 2^{14} of the IOP accumulator into bits 2^9 through 2^{23} of Central Memory starting address register. For an SSD, this function enters bits 2^0 through 2^{14} of the IOP accumulator into the 2^{15} through 2^{29} bit positions of the SSD starting address register. The states of the Busy and Done flag are not altered. This function causes an error condition if issued when the output channel is active.

****

CAUTION: Unused bits must be zeroed.

****

HOA : 3 - ENTER LOWER CENTRAL MEMORY OR SSD ADDRESS

This function enters bits 2^0 through 2^8 of the IOP accumulator into bits 2^0 through 2^8 of the Central Memory starting address register. For an SSD, bits 2^0 through 2^8 of the IOP accumulator enter bit positions 2^6 through 2^{14} of the SSD starting address register. The states of the Busy and Done flags are not altered. This function causes an error condition if issued when the output channel is active.

****

CAUTION: Unused bits must be zeroed.

****

NOTE

The 22-bit SSD address entered through the HOA : 2 and HOA : 3 functions is the SSD starting address divided by 100_8.

HOA : 5 - WRITE CENTRAL MEMORY OR SSD, ENTER BLOCK LENGTH

This function enters bits 2^0 through 2^{13} of the IOP accumulator into the Block Length register. (This value is treated as a count of 64-bit words with a zero value indicating the maximum 16,384 words.) The Busy flag is set, the Done flag is cleared, and the transfer is initiated. Upon completion, the Done flag sets and the Busy flag clears. If during the transfer an unrecoverable error occurs, the transfer terminates, the Done flag sets, and the Busy flag stays set. This function causes an error condition if issued while the output channel is active.

NOTE

When connected to an SSD, bits 2^0 through 2^5 of the word count must be zeros to avoid a block length error.

HOA : 6 - CLEAR INTERRUPT ENABLE

This function clears the Interrupt Enable flag. The states of the Busy and Done flags are not altered.

HOA : 7 - SET INTERRUPT ENABLE

This function sets the Interrupt Enable flag. The states of the Busy and Done flags are not altered.

HOA : 14 - ENTER DIAGNOSTIC MODE

This function enters bits 2^0 through 2^2 of the IOP accumulator into the Diagnostic Mode register. The modes are summarized in table 7-11. This function is only for maintenance purposes.

With the exception of mode 6, only one mode is valid at a time. Mode 6 is held if it has been set and cannot be cleared by entering another mode. All diagnostic modes clear with Master Clear, or alternately by doing HOA : 14, with the accumulator = 0, followed by HOA : 0. Mode 0 is active until any other mode is selected.

Table 7-11. Output Channel Diagnostic Modes

Mode Designator	Function
0	Set Last Word flag signal (only if channel is active)
1	Disable Last Word flag signal to Central Memory or SSD
2	Force constant Address Ready
3	Disable third Address Ready
4	Transfer first address with zero parity bits, third address with one parity bits, and disable Data Ready signal to Central Memory or SSD
5	Force Clear Channel signal without inactivating IOP
6	Use bits 2^{56} through 2^{63} of the first word as the second word check byte, bits 2^{56} through 2^{63} of the third word as fourth word check byte, and so on
7	Disable error correction and detection at Central Memory or disable reporting unrecoverable data channel errors by SSD input channel. SSD memory errors are still reported to SSD error logger.

CENTRAL MEMORY OR SSD OUTPUT ERROR PROCESSING

If an unrecoverable error occurs in an output transfer to Central Memory or SSD, the Busy and Done flags both set. An error code is generated and sent to an error log. The error codes are summarized in table 7-12.

The only recoverable error is a corrected single-bit data error. This type of error is ignored by the channel interface because it is corrected automatically. The syndrome information is sent to an error log.

Table 7-12. Output Channel Error Codes

Error Code	Name	Condition
1	Function Error	A function 0, 1, 2, 3, or 5 was issued while the channel was active.
2	Active Error	Central Memory or SSD side went inactive while the IOP side was still active.
3	Transmit Address Time out	An IOP output was initiated, but Central Memory or SSD did not send a Transmit Address signal within 2 ms.
4	Central Memory or SSD Address Error	Central Memory or SSD side received greater or less than three Address Ready signals, a parity error occurred in one of the address channel transfers, or Central Memory or SSD received data while active.
5	Central Memory or SSD Data Block Length Error	Either Central Memory or SSD received data with a multiple error, the last word was received and the block length count was not 0, or the last word was not received and the block length count was equal to 0.
6	Last Word Time-out	The last word was sent, but Central Memory or SSD did not go inactive within 2 ms.
7	Transmit Data Time-Out	A Transmit Data signal was not received within 2 ms.

MEMORY (100 MBYTE) CHANNEL OUTPUT SEQUENCE

Figure 7-5 shows the sequence of signals for transferring data from Local Memory to Central Memory or an SSD.

ERROR LOGGING CHANNEL (Serial No. 20 and below)

On I/O Substems Serial No. 20 and below, an IOP may have an error logging channel connected. This channel reports errors from the Local Memories of three other I/O Processors, from Buffer Memory, from Central Memory, and from the Memory (100 Mbyte) Channels to Central Memory. If any error condition occurs in any of the reporting devices, the Channel Done flag sets. The functions are listed below. (See page 7-47 for information concerning error logging in machines with serial numbers 21 and up.)

Function	Description
ERA : 0†	Idle channel
ERA : 6††	Clear Interrupt Enable flag
ERA : 7††	Set Interrupt Enable flag
ERA : 10	Read error status
ERA : 11	Read error information (first parameter)
ERA : 12	Read error information (second parameter)
ERA : 13	Read error information (third parameter)

INTERFACE REGISTERS

The error log interface has four interface registers. The error status register is a 9-bit register that holds set bits for the device causing the error. This register is accessed by the ERA : 10 command.

The first parameter word register has 16 bits and stores Local Memory failing locations, or Buffer Memory error data, or Memory (100 Mbyte) Channel error data.

The second parameter word register has 16 bits and stores the low-order bits of the address for the Buffer Memory or Central Memory failing locations.

The third parameter word register has 8 bits, and stores the high-order bits of the address for either Buffer Memory or Central Memory.

---

† Allow 1 CP before checking Busy or Done flags.
†† Allow 1 CP before checking the interrupt channel number (IOR : 10).

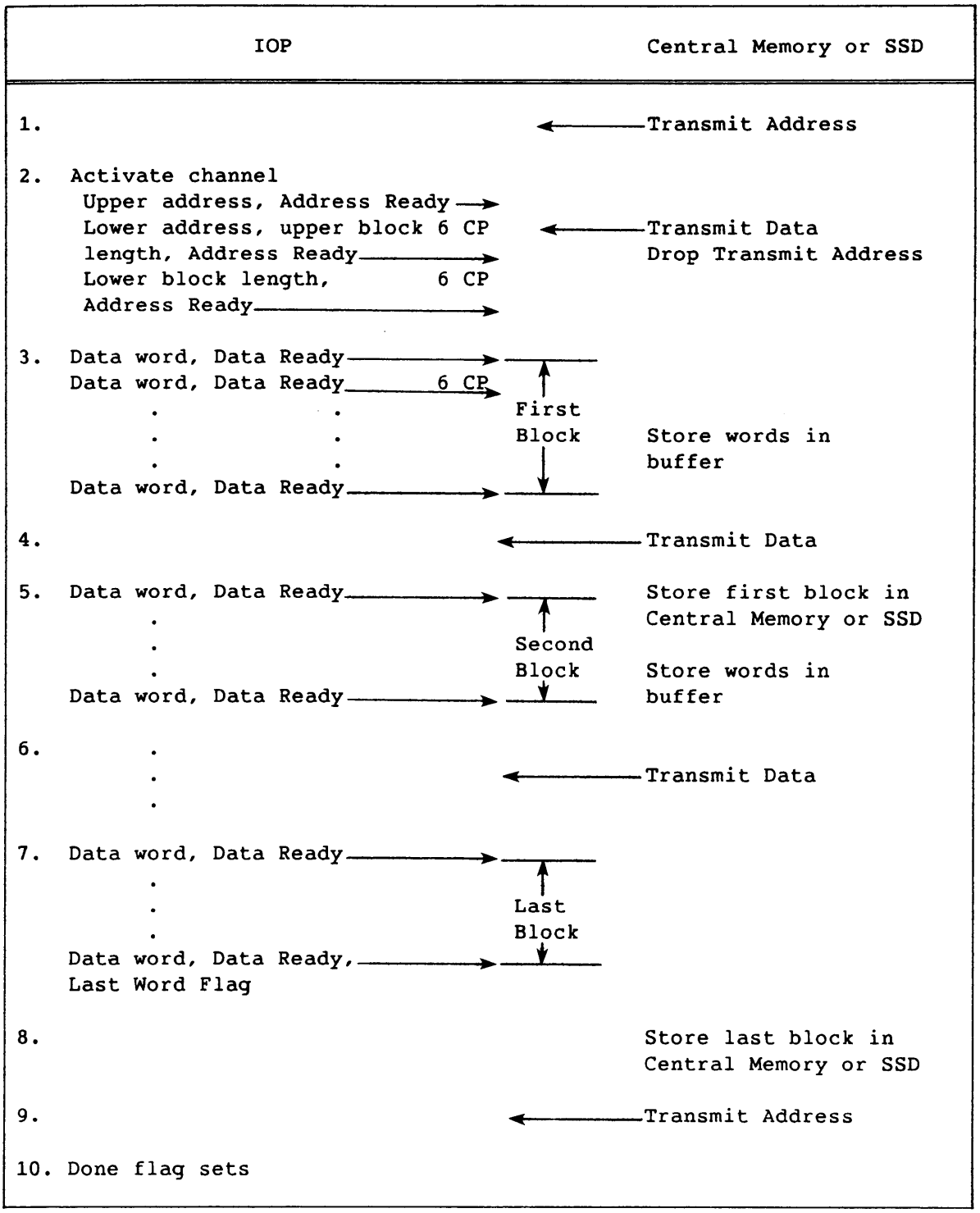


Figure 7-5. Memory (100 Mbyte) Channel Sequence on Output from I/O Processor

ERA : 0 - IDLE CHANNEL

This function clears all error flags stored in the interface registers. The Done flag is cleared.

ERA : 6 - CLEAR INTERRUPT ENABLE FLAG

This function clears the Interrupt Enable flag for this channel.

ERA : 7 - SET INTERRUPT ENABLE FLAG

This function enables the interface to interrupt the IOP.

ERA : 10 - READ ERROR STATUS

This function returns the contents of the interface Error Status register to the accumulator.

NOTE

Function ERA : 11 must follow function ERA : 10 if functions ERA : 12 and ERA : 13 (second and third parameters) are read.

The interpretation of the set bits is given in table 7-13.

The selection of physical devices such as I/O Processors and Memory (100 Mbyte) Channels as A, B, C, and so on is part of the overall system definition. Appendix C has a typical system definition.

Errors occurring in an IOP having the error logging channel are reported to that IOP through its own Local Memory error (LME) channel.

Table 7-13. Error Status Register Bits

Bit	Control Signal
2^0	IOP-1 Local Memory error
2^1	IOP-2 Local Memory error
2^2	IOP-3 Local Memory error
2^3	Buffer Memory error
2^4	Central Memory error
2^5	Memory (100 Mbyte) Channel input A error
2^6	Memory (100 Mbyte) Channel output B error
2^7	Memory (100 Mbyte) Channel input C error
2^8	Memory (100 Mbyte) Channel output D error

ERA : 11 - READ ERROR INFORMATION (FIRST PARAMETER)

This function returns an error parameter word to the accumulator, selected by a bit set in the accumulator when the function issues. Only one bit among the 2^0 through 2^8 positions can be set at one time. The accumulator bits select parameters as listed in table 7-14.

When an IOP Local Memory address or Memory (100 Mbyte) Channel error data word is read, the device bit is cleared in the Status register.

The Address Invalid flag (2^{15}) in the Buffer Memory error data is set when the address that will be given in the second and third parameter words is not the correct address for the error being reported in the first parameter word. This situation can occur when a later read reference follows closely after a read reference that detects an error in its data. The address for the later read reference is incorrectly reported in the second and third parameter words and should be disregarded.

NOTE

Function ERA : 11 must be immediately preceded by a logical product instruction as follows:

```
011777  A=A&d  d=777
171xxx  ERA : 11
        or
015xxx  A=A&k
xxx777  k=xxx777
171xxx  ERA : 11
```

Table 7-14. First Error Parameter Selection

Register Bit	First Parameter Information Returned
2^0	IOP-1 Local Memory failing address: 2^0 - 2^1 Bank number 2^2 - 2^3 Section number 2^4 Byte: 0 Bits 2^0 through 2^7 1 Bits 2^8 through 2^{15}
2^1	IOP-2 Local Memory failing address (same format as for 2^0)
2^2	IOP-3 Local Memory failing address (same format as for 2^0)
2^3	Buffer Memory error data: 2^0 - 2^7 Syndrome bits 2^9 Correctable error if 1 2^{10} Uncorrectable error if 1 2^{12} - 2^{13} Port number 2^{15} Address invalid if 1
2^4	Central Memory error data: 2^0 - 2^7 Syndrome bits 2^9 Correctable error 2^{10} Uncorrectable error 2^{12} - 2^{13} Read mode: 0 Scalar 1 I/O 2 Vector 3 Fetch
2^5	Input Memory (100 Mbyte) Channel A error data: 2^0 - 2^7 Syndrome bits 2^8 - 2^{10} Error code (refer to Central Memory input error processing in this section)
2^6	Output Memory (100 Mbyte) Channel B error data, same as for 2^5. Refer to Central Memory output error processing in this section for error codes.
2^7	Input 100 Mbyte Channel C error data, same format as for 2^5
2^8	Output 100 Mbyte Channel D error data, same format as for 2^5. Refer to Central Memory output error processing in this section for error codes.

ERA : 12 - READ ERROR INFORMATION (SECOND PARAMETER)

This function returns the contents of the second parameter word register to the accumulator. The second parameter word is selected by bits 2^3 or 2^4 in the accumulator when the function is sent. Only 1 bit can be set at a time in the accumulator 2^3 and 2^4 bit positions. Selection is as follows:

Bit	Meaning
2^3	Indicates the low-order bits of the Buffer Memory address (address bits 2^0 through 2^{15})
2^4	Indicates the low-order bits of the Central Memory address (address bits 2^0 through 2^{15})

NOTE

Function ERA : 12 must be immediately preceded by a logical product instruction as follows:

```
011777  A=A&d   d=777
172xxx  ERA : 12

        or

015xxx  A=A&k
xxx777  k=xxx777
172xxx  ERA : 12
```

ERA : 13 - READ ERROR INFORMATION (THIRD PARAMETER)

This function returns the contents of the third parameter word register to the accumulator. The third parameter word is selected by bits 2^3 or 2^4 in the accumulator when the function issues. Only 1 bit can be set at a time in the field. Selection is as follows:

Bit	Meaning
2^3	Returns the high-order bits of the Buffer Memory address (bits 2^{16} through 2^{22} going into accumulator positions 2^0 through 2^6)
2^4	Returns the high-order bits of the Central Memory address (address bits 2^{16} - 2^{21} going into accumulator positions 2^0 through 2^5)

The reading of the third parameter word clears the error bits in the Status register for the Central Memory or for the Buffer Memory.

NOTE

Function ERA : 13 must be immediately preceded by a logical product instruction as follows:

```
011777  A=A&d   d=777
173xxx  ERA : 13

        or

015xxx  A=A&k
xxx777  k=xxx777
173xxx  ERA : 13
```

ERROR LOGGING CHANNEL (Serial No. 21 and higher)

IOSs with serial numbers 21 and higher use an error multiplex for detecting and reporting IOP subsystem errors. This multiplex passes channel error information and memory error information to a maintenance computer where the maintenance computer program logs the error information for later analysis.

The multiplex module captures single and multiple errors, even if the multiple error is embedded in a burst of single-bit errors. Channels that are multiplexed include the Buffer Memory channel, up to four Local Memory channels, and two high-speed channels. Additional multiplex modules are added if necessary to provide error logging of additional high-speed channels (up to 10).

Errors are not available to the Master I/O Processor (MIOP) in Serial No. 21 and higher machines. The IOP is still aware of any catastrophic error (DBE or Local Memory) through its Local Memory Error channel, or through the Buffer Memory or Memory Channel Busy and Done flags. All detailed information, however, is sent to the external error logger.

BLOCK MULTIPLEXER CHANNEL

The I/O Subsystem communicates with IBM-compatible equipment through the XIOP over the block multiplexer channels. The following functions are used.

Function	Description
BMA : 0	Clear channel control
BMA : 1	Send reset function
BMA : 2	Send device command
BMA : 3	Read Request-in address
BMA : 4	Asynchronous I/O
BMA : 5	Delay counter diagnostic
BMA : 6†	Clear Channel Interrupt Enable flag
BMA : 7†	Set Channel Interrupt Enable flag
BMA : 10	Read Local Memory address
BMA : 11	Read byte counter status
BMA : 12	Read status/address
BMA : 13	Read input tags
BMA : 14	Enter Local Memory address
BMA : 15	Enter byte count
BMA : 16	Enter device address/mode
BMA : 17	Enter output tags

† Allow 1 CP before checking the interrupt channel number (IOR : 10).

GENERAL CHARACTERISTICS

Block multiplexer channels are grouped into sets of four channels, which share one Cray Research, Inc. BMC-4 Block Multiplexer Controller. The controller interfaces to the Auxiliary I/O Processor (XIOP) through one direct memory access (DMA) port. The speed capacity of the DMA port is shared among the four controller channels. The four channels operate asynchronously and compete only for Local Memory references.

A block multiplexer channel operates in either selector channel mode, byte multiplexer mode, or block multiplexer mode. All IBM commands are possible. The channel features command and data chaining, detection of retry status, and the I/O error alert and the high speed option.

The channel drives one or more IBM-compatible control units, which in turn drive peripheral devices.

This manual does not explain or document the IBM communications protocol, the signals used, nor their sequencing. Refer to the appropriate IBM publications for detailed information and definitions of IBM terminology.

TRANSFER RATES

The block multiplexer channel data rate is determined by cable length and signal turnaround time within the control unit. For each byte of data or control information that is sent, an appropriate response signal or signals must be received. The resulting data rate limit is about 6.4 million bits a second, or 800 kilobytes a second. IBM-compatible peripheral controllers that use the high speed option (requiring an additional pair of control lines) can achieve about 12.8 million bits a second, or 1.6 million bytes a second.

DATA HANDLING

One important feature of the channel is the assembly and disassembly of data between the 8-bit IBM channel and the 16-bit memory parcels. Figure 7-6 shows the data changes.

As shown in figure 7-6, the BMC-4 reads four 16-bit parcels from Local Memory and sends out eight 8-bit bytes to the block multiplexer channel. Four more 16-bit parcels are read into buffers in the controller while the channel is transmitting the first group.

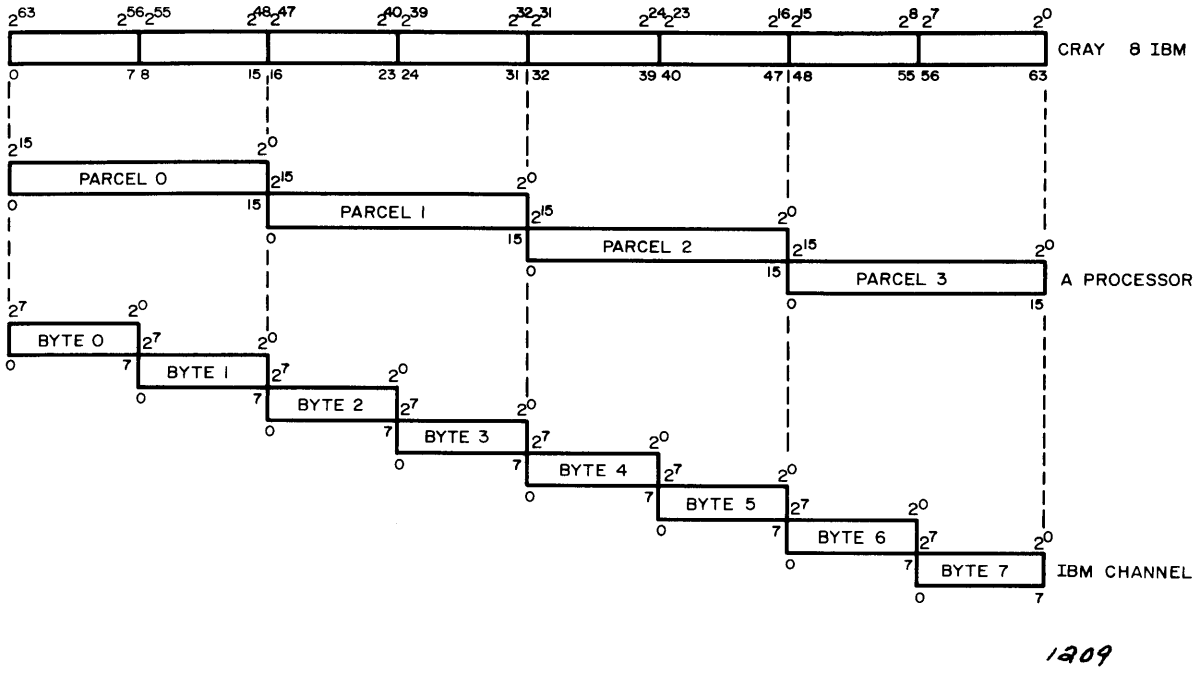


Figure 7-6. BMC-4 Data Assembly/Disassembly

The BMC-4 reads eight 8-bit bytes from the channel and writes four 16-bit parcels into Local Memory. If a read operation from the channel terminates with a byte length that does not evenly comprise 4 parcels, the last Local Memory write includes unpredictable data in the last parcels.

RECORD SIZE

Data records can be any nonzero integer number of bytes in length. However, the data chaining feature must be used for lengths greater than 65,535 bytes.

In cases such as IBM tape records, an odd length header field can be read and stored at one memory area and the following data record can be stored at a different memory area beginning at a Buffer Memory word boundary. The data chaining feature permits a single large record to be broken into any size convenient for storage in Buffer Memory. This process is under program control by means of I/O functions to the block multiplexer channel.

PARITY

Odd parity is checked on all address, status, and data inputs to the IOP and is generated for all address, control, or data outputs from the IOP.

INTERRUPTS

All interrupts can be enabled or disabled for any block multiplexer channel. If interrupts are enabled, and a function in the range BMA : 1 through BMA : 5 completes, an interrupt request is set. If an interrupt is selected for an input tag line such as request in, the interrupt request sets when the request-in line goes to a logical 1.

When data chaining is enabled, an interrupt request is set each time the byte counter decrements to 0 and memory references are complete through that particular block of data.

BMA : 0 - CLEAR CHANNEL BUSY AND DONE FLAGS

This function clears the Channel Busy and Done flags and all output tags except OP-OUT and SUP-OUT. No parameters are required for this action. This function also clears interrupt conditions, provided interrupts are disabled by a BMA : 6 function. Since this function cannot be interlocked using the Channel Done flag, allow a 12-CP delay before issuing the next function to that particular channel.

BMA : 1 - SEND RESET FUNCTION

Several reset functions (table 7-15) perform various functions required by the equipment. Bits 2^1 and 2^0 of the accumulator contents are used as a parameter that selects the specific function.

Table 7-15. Send Reset Function Parameters

Parameter 2^1 - 2^0	Function
0 0	Clear all output tag lines
0 1	Interface disconnect
1 0	Selective reset
1 1	System reset

Parameter *xxxxx*0 - Clear output tag lines

Parameter *xxxxx*0 clears all output tag lines and clears BUS 0 Out lines. The channel initially clears the Channel Done flag and sets the Channel Busy flag. Upon completion, the Channel Done flag sets and the Channel Busy flag clears.

Parameter *xxxxx*1 - Interface disconnect

Parameter *xxxxx*1 performs the interface disconnect function to the currently selected control unit. The function clears the Channel Done flag and sets the Channel Busy flag. The control unit removes all signals from the IOP channel. When the control unit reaches the normal ending point in its sequence, it attempts to obtain selection in order to present any generated status to the channel. The control unit does not generate any status as a result of the interface disconnect.

The device path for the peripheral remains busy after it receives an interface disconnect during an operation until the device-end status is accepted by the block multiplexer channel.

The function should complete in about 7 microseconds. At that time the IOS Channel Done flag sets and the Channel Busy flag clears.

If the operational-in signal from the equipment does not clear within 6 microseconds from the function issue, the IOS Channel Done flag sets and the Channel Busy flag remains set, indicating the error condition.

Parameter *xxxxx*2 - Selective reset

Parameter *xxxxx*2 performs the selective reset function. The Channel Done flag clears and the Channel Busy flag sets. The function causes the operational-in signal to clear and resets the currently selected peripheral device along with its status. The current operation proceeds to a normal stopping point and no data is transferred after the stop.

Only the peripheral device currently operating is affected. The device-end status is retained for transfer to the IOP channel after the reset.

The ready or not ready state of the control unit is generally not changed by the selective reset function.

The function should complete in about 7 microseconds, at which time the IOS Channel Busy flag clears and the Channel Done flag sets.

Parameter xxxxx3 - System reset

The system reset function clears the operational-out signal and resets all control units along with their attached peripheral devices. The status for each peripheral device is also reset. The Channel Done flag clears and the Channel Busy flag sets at the beginning of execution.

The function should complete in about 6 microseconds at which time the IOP Channel Done flag sets and the Channel Busy flag clears. Input tag lines that are not reset by the system reset function cause the Channel Busy flag to remain set when the Channel Done flag sets.

BMA : 2 - CHANNEL COMMAND

This function sends commands to the control units. The specific command is selected by the accumulator bits at the time the function issues. Many of the commands require prior functions to set up interface registers, such as the Local Memory Address register, the Byte Count register, and the Device Address register. The command parameter bits are shown in table 7-16. The Channel Done flag clears and the Channel Busy flag sets at the beginning of execution. At completion of the function, the Channel Busy flag clears and the Channel Done flag sets.

Table 7-16. Channel Command Function Parameter Bits

Parameter Bit	Assignment
2^0	IBM-type control unit command bit 2^0
2^1	IBM-type control unit command bit 2^1
2^2	IBM-type control unit command bit 2^2
2^3	IBM-type control unit command bit 2^3
2^4	IBM-type control unit command bit 2^4
2^5	IBM-type control unit command bit 2^5
2^6	IBM-type control unit command bit 2^6
2^7	IBM-type control unit command bit 2^7
2^8	Unused
2^9	Unused
2^{10}	Unused
2^{11}	Unused
2^{12}	Unused
2^{13}	Unused
2^{14}	Unused
2^{15}	Unused

Parameter command bits

Bits 2^0 through 2^7 are the command bits for the BMA : 2 parameter. The command is for an IBM-type control unit. The parameter bits are shown in table 7-17.

Table 7-17. Channel Command Bit Assignments

Command	P	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Test I/O	1	0	0	0	0	0	0	0	0
Sense	P	M	M	M	M	0	1	0	0
Read backward	P	M	M	M	M	1	1	0	0
Write	P	M	M	M	M	M	M	0	1
Read	P	M	M	M	M	M	M	1	0
Control	P	M	M	M	M	M	M	1	1

M = Modifier bit
P = Parity bit

Specific modifier codes and the particular mode set depend on the IBM-type control unit and the peripheral device used. The command byte is sent only during the initial selection sequence. All commands except test I/O can require a data transfer to satisfy the function.

All commands begin with an initial selection sequence which ends with either an ending status (channel-end or channel-end with device-end), a zero status, or an error status. To a command calling for transfer of status, control or data bytes, a zero status signifies that the transfer can begin.

A data-type transfer ends when the byte count decrements to 0 and no data chaining condition exists. The data-type transfer also ends when status-in is received in response to service-out or data-out.

The Channel Done flag is not set until processing of the control sequence is completed and data is transferred to or from Local Memory.

Figure 7-7 shows a typical channel sequence for a read to the I/O Processor.

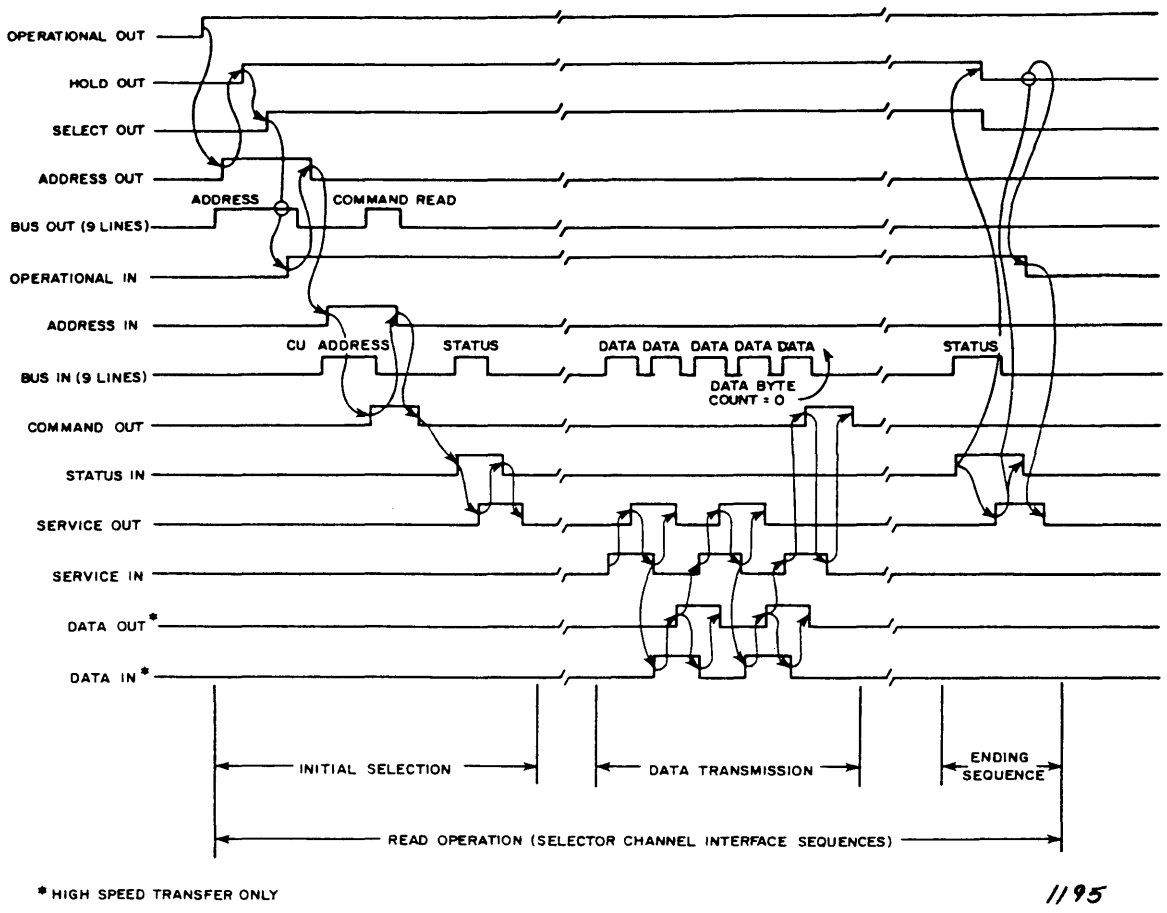


Figure 7-7. Channel Read Sequence

BMA : 3 - READ REQUEST-IN ADDRESS

Function BMA : 3 first clears the Channel Done flag and sets the Channel Busy flag. If request-in is detected, the channel accepts the requesting address. The address is checked for valid parity, the Channel Done flag is set, and the Channel Busy flag is cleared.

If request-in is not detected, the Channel Done flag is set and the Channel Busy flag stays set. The address miscompare status is to be ignored if undetermined addresses are expected.

Figure 7-8 illustrates a typical request-in channel sequence.

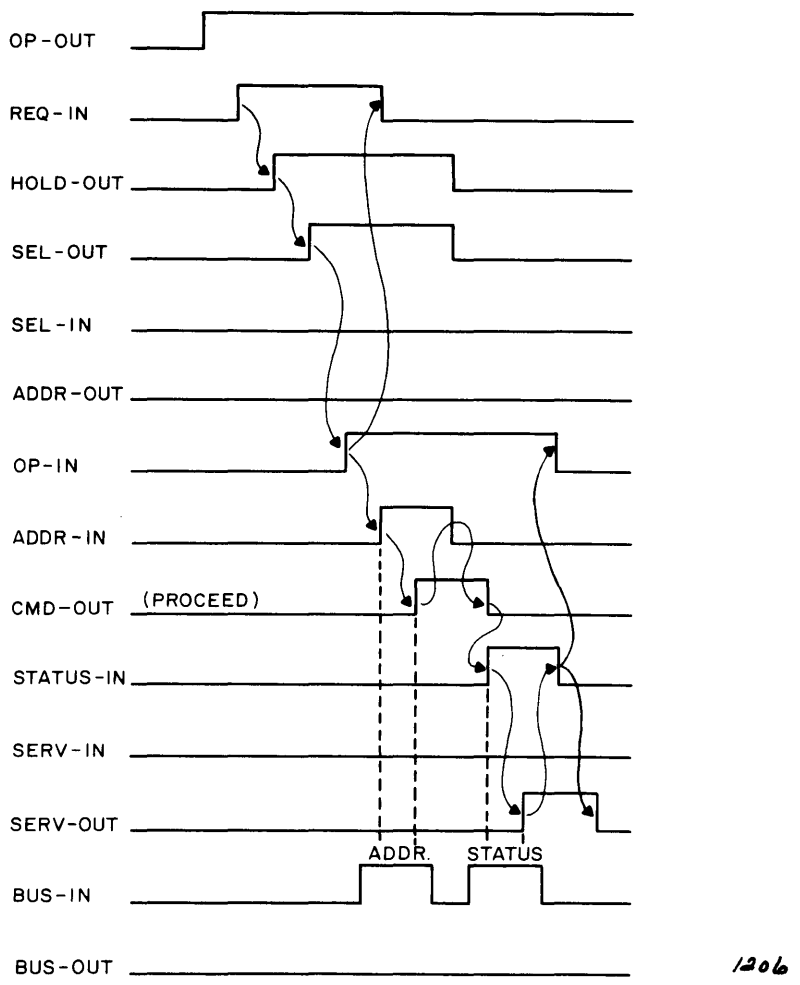


Figure 7-8. Channel I/O Sequence

BMA : 4 - ASYNCHRONOUS I/O

Issuing the BMA : 4 function clears the Channel Done flag and sets the Channel Busy flag. If status-in is present, status is saved in the Status register. If the Stack Status flag has been set (by the last BMA : 16 command), command-out is returned in place of service-out to indicate stop to the channel. When the sequence completes, the Channel Done flag sets and the Channel Busy flag clears. Refer to figure 7-8 for the typical request-in channel sequence. See figure 7-9 for the asynchronous data and status processing. See table 7-16 for accumulator parameter bits 2^0 through 2^7.

NOTE

If asynchronous data and status processing takes place in interrupt mode, request-in must be cleared and disabled before issuing the IOB : 3 function as follows:

```
IOB : 6          .Clear request-in
A = 0
IOB : 16         .Disable request-in
IOB : 0          .Clear interrupt
```

A constant interrupt occurs if this procedure is not followed.

BMA : 5 - DELAY COUNTER DIAGNOSTIC

Function BMA : 5 clears the Channel Done flag and sets the Channel Busy flag. The channel then performs a 10-microsecond delay. After the delay times out, the Channel Done flag sets and the Channel Busy flag clears. Function BMA : 5 is for maintenance purposes only.

BMA : 6 - CLEAR CHANNEL INTERRUPT ENABLE FLAG

Function BMA : 6 clears the Channel Interrupt Enable flag. This prevents the channel from interrupting the IOP. In this case, the IOP monitors the Channel Done flag to determine function completion. The Channel Busy and Channel Done flags are not affected by this function.

BMA : 7 - SET CHANNEL INTERRUPT ENABLE FLAG

Function BMA : 7 sets the Channel Interrupt Enable flag. Setting this flag causes an IOP interrupt for this channel whenever any of the following conditions occur:

- Channel Done flag sets
- Interrupt mode select, through a BMA : 16 function, has enabled an interrupt for an active input tag line
- During data chaining, the byte counter has decremented to 0 and the last Local Memory reference is complete for that segment of data

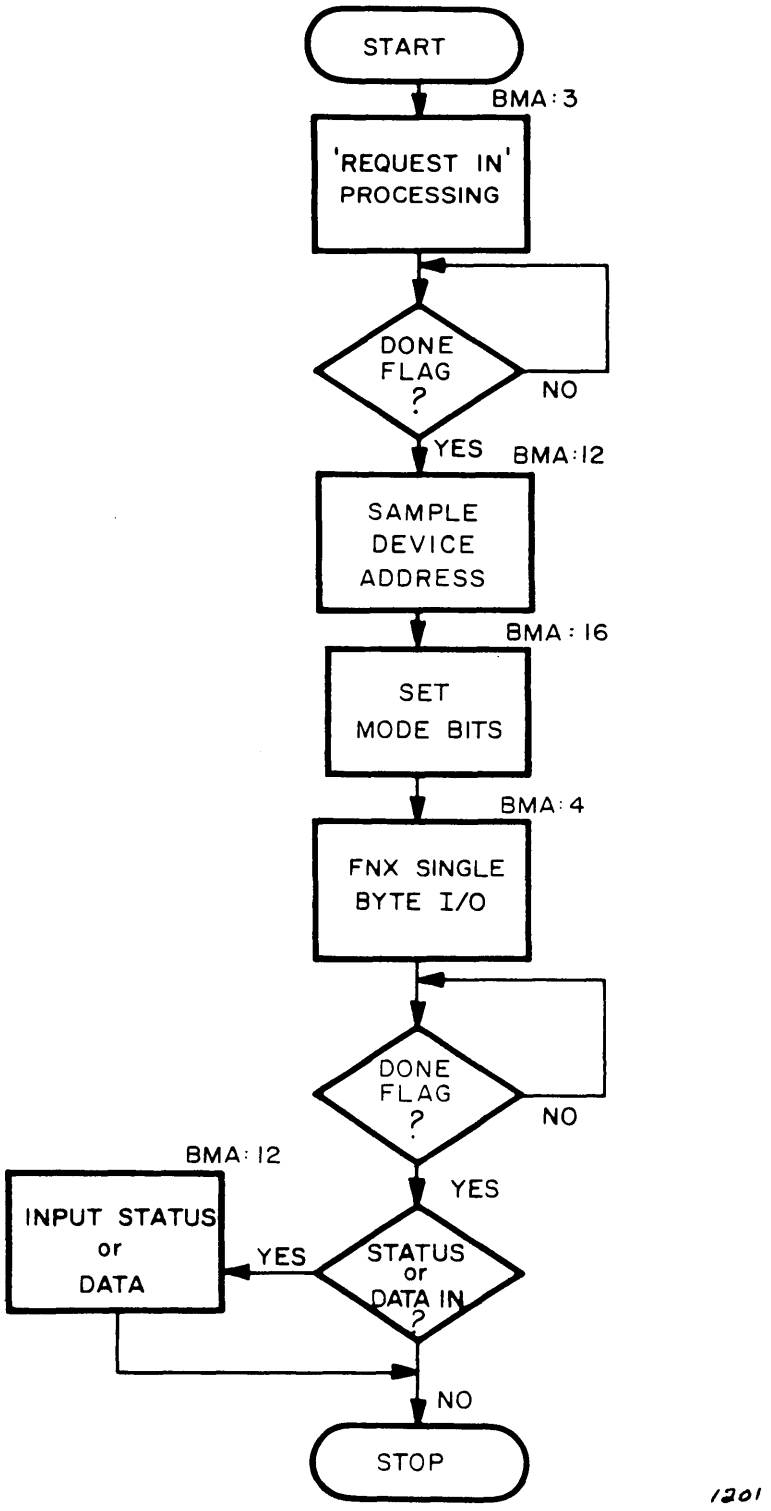


Figure 7-9. Asynchronous Data and Status Processing

BMA : 10 - READ LOCAL MEMORY ADDRESS

Function BMA : 10 reads the current value in the Local Memory Address register and enters the value in the accumulator. The channel logic includes two Local Memory Address registers to support data chaining. The state of accumulator bit 2^0 when the function is issued determines which Local Memory Address register is to be read. Bit 2^0 of the value returned to the accumulator identifies the register from which the Local Memory address came.

Bit 2^1 in the value returned to the accumulator is set if data chaining is being used. Table 7-18 lists the accumulator contents resulting from function BMA : 10.

Function BMA : 10 can be performed at any time relative to control functions.

NOTE

When a BMA : 14, 15, 16, or 17 function is to be followed immediately by a BMA : 10 or 11 function, a single instruction delay (012000_8) must be inserted between the functions to ensure that the Local Memory Address returned will be for the expected channel.

During data chaining, interrupts occur after each buffer of data is transferred. Function BMA : 10 clears the interrupt after a data transfer.

Function BMA : 10 has a programming restriction due to timing in the Adder and Shifter. The restriction applies if the instruction preceding a BMA : 10 function is any of the following: 4 through 7, 12, 13, 16, 17, 22, 23, 32, 33, 44 through 47, 52, 53, 62, or 63. In these cases a logical product instruction 011 or 015 with the *d* or *k* fields set to all ones should be inserted between the above instruction and the BMA : 10 function.

BMA : 11 - READ BYTE COUNTER

The byte counter records the number of bytes remaining in a data transfer. This counter is initially loaded with a value by a BMA : 15 load byte counter function. When the counter decrements byte-by-byte to 0, the channel interrupts the IOP and terminates the transfer. A transfer terminated by the control unit and not by decrementing to 0 can result in a nonzero value in the counter. If data chaining is requested, the byte counter is reloaded after decrementing to 0.

Table 7-18. Read Local Memory Address Response Bits

Accumulator Bit	Meaning
2^0	Register select status
2^1	Data chaining flag status (1=data chaining)
2^2	Local Memory address 2^2
2^3	Local Memory address 2^3
2^4	Local Memory address 2^4
2^5	Local Memory address 2^5
2^6	Local Memory address 2^6
2^7	Local Memory address 2^7
2^8	Local Memory address 2^8
2^9	Local Memory address 2^9
2^{10}	Local Memory address 2^{10}
2^{11}	Local Memory address 2^{11}
2^{12}	Local Memory address 2^{12}
2^{13}	Local Memory address 2^{13}
2^{14}	Local Memory address 2^{14}
2^{15}	Local Memory address 2^{15}

Due to the fixed timing in the IOP, this function must be executed twice in succession to get a current value byte counter status to the IOP accumulator. The first execution moves the current byte counter status to the block multiplexer controller. The second execution moves the status to the IOP accumulator.

Function BMA : 11 can be used to verify the accumulator fanout, the intermediate Byte Counter Status register in the block multiplexer controller, and the status path back to the IOP accumulator. Issuing a BMA : 15 function (explained later in this section) loads the byte count into the Byte Counter Status register in the block multiplexer controller. The next BMA : 11 function reads the byte count back from the Status register to the IOP accumulator. The second BMA : 11 function performs as described above.

NOTE

When a BMA : 14, 15, 16, or 17 function is to be followed immediately by a BMA : 10 or 11 function, a single instruction delay (012000_8) must be inserted between the functions to ensure that the Local Memory Address returned will be for the expected channel.

The Channel Busy and Channel Done flags are not affected by this function.

BMA : 12 - READ STATUS/ADDRESS

The Status register holds the address and status mode bits read from the block multiplexer channel.

Due to the fixed timing in the IOP, function BMA : 12 must be executed twice in immediate succession to get valid current status/address information to the IOP accumulator. The first execution moves the current status/address to the block multiplexer controller. The second execution moves the status/address to the IOP accumulator.

This function can verify the accumulator fanout, the intermediate Status register in the block multiplexer controller, and the status path back to the IOP accumulator. Issuing a BMA : 16 function (explained later in this section) loads address and mode bits into the block multiplexer controller status register. The next BMA : 12 function reads the status back to the IOP accumulator. The second BMA : 12 function performs as described above.

The Channel Busy and Channel Done flags are not affected by this function.

The status/address returned is the status/address information from the channel and is entered in the IOP accumulator as shown in table 7-19.

Table 7-19. Status Register Bits

Accumulator Bit	Meaning
2^0	Status 2^0
2^1	Status 2^1
2^2	Status 2^2
2^3	Status 2^3
2^4	Status 2^4
2^5	Status 2^5
2^6	Status 2^6
2^7	Status 2^7
2^8	Address 2^0
2^9	Address 2^1
2^{10}	Address 2^2
2^{11}	Address 2^3
2^{12}	Address 2^4
2^{13}	Address 2^5
2^{14}	Address 2^6
2^{15}	Address 2^7

BMA : 13 - READ INPUT TAGS

Function BMA : 13 reads the input tags from the channel to the IOP accumulator.

Due to the fixed timing in the IOP, this function must be executed twice in immediate succession to get valid current input tags to the IOP accumulator. The first execution moves the current input tags to the block multiplexer controller. The second execution moves the input tags to the IOP accumulator.

This function can verify the accumulator fanout, the intermediate output tags register, and the path back to the accumulator. Issuing a BMA : 17 function (explained later in this section) loads the output tags into the output tags register. The next BMA : 13 function reads the stored output tags back to the IOP accumulator. The second BMA : 13 function brings the current input tags to the IOP accumulator.

The Channel Busy and Channel Done flags are not affected by this function.

The input tags status bits are shown in table 7-20.

Table 7-20. Input Tags Status Bits

Accumulator Bit	Meaning
2^0	Operational in
2^1	Address in
2^2	Disconnect in
2^3	Select in
2^4	Request in
2^5	Service in
2^6	Data in
2^7	Status in
2^8	Metering in
2^9	Mark 0 in
2^{10}	Unused (0)
2^{11}	Data buffer pointer
2^{12}	Address miscompare
2^{13}	Byte count 0
2^{14}	P-ROM parity error
2^{15}	BUS-IN parity error

BMA : 14 - ENTER LOCAL MEMORY ADDRESS

Function BMA : 14 enters the current accumulator contents into the Local Memory Address register. This address is the starting address in Local Memory for the data transfer. The Channel Busy and Channel Done flags are not altered in the process. Two Local Memory Address registers are maintained for data chaining purposes, and they are addressed by the 2^0 bit of the accumulator contents. Data chaining must begin with register 0 and alternate between the 0 and 1 registers. Bit 2^1 of the accumulator contents is the data chaining select flag for the chosen register and, if set, selects data chaining for that register/address. Table 7-21 lists the accumulator bits for this function.

BMA : 15 - ENTER BYTE COUNT

Function BMA : 15 enters the accumulator contents into the byte counter or into the next byte count register. The first BMA : 15 function following a BMA : 14 enters the accumulator contents into the byte counter. Immediately following the first BMA : 15 function with a second BMA : 15 function enters the accumulator data into the next byte counter register.

Table 7-21. Local Memory Address Register Bits

Accumulator Bit	Meaning
2^0	Local Memory Address register file select
2^1	Data chaining flag
2^2	Local Memory address 2^2
2^3	Local Memory address 2^3
2^4	Local Memory address 2^4
2^5	Local Memory address 2^5
2^6	Local Memory address 2^6
2^7	Local Memory address 2^7
2^8	Local Memory address 2^8
2^9	Local Memory address 2^9
2^{10}	Local Memory address 2^{10}
2^{11}	Local Memory address 2^{11}
2^{12}	Local Memory address 2^{12}
2^{13}	Local Memory address 2^{13}
2^{14}	Local Memory address 2^{14}
2^{15}	Local Memory address 2^{15}

The transfer from the next byte count register to byte counter is done automatically between data segments in data chaining. Channel commands requiring no data, parameters, or status must establish a byte count of 0. The maximum count is 65,535 bytes. The Channel Busy and Channel Done flags are not altered in this process.

BMA : 16 - ENTER DEVICE ADDRESS/MODE

Function BMA : 16 enters the accumulator contents into the device address register. The device address register contains mode select bits and device address bits. The device address can be some combination of controller address bits and peripheral device address bits. The Channel Busy and Channel Done flags are not altered by this function. Table 7-22 shows the device address register bits.

Parameter mode bits

Bits 2^8 through 2^{15} of the parameter are called mode bits and are reestablished during each BMA : 16 function. They operate as follows.

Skip flag - Bit 2^8 is the mode bit for the Skip flag. When set, it prohibits storing data into Local Memory during read data transfers.

Table 7-22. Device Address Register Bits

Accumulator Bit	Meaning
2^0	Address/data out 2^0
2^1	Address/data out 2^1
2^2	Address/data out 2^2
2^3	Address/data out 2^3
2^4	Address/data out 2^4
2^5	Address/data out 2^5
2^6	Address/data out 2^6
2^7	Address/data out 2^7
2^8	Skip flag
2^9	Stack status flag
2^{10}	Command chaining mode select 2^0
2^{11}	Command chaining mode select 2^1
2^{12}	Interrupt mode select 2^0
2^{13}	Interrupt mode select 2^1
2^{14}	Channel mode select 2^0
2^{15}	Channel mode select 2^1

Stack Status flag - Bit 2^9 is the mode bit for the Stack Status flag.

Command chaining mode select - Bits 2^{10} and 2^{11} select the mode in which command chaining will be used. Table 7-23 shows the translation of these mode bits.

Interrupt mode select - Parameter bits 2^{12} through 2^{13} select the mode in which interrupts are generated. Table 7-24 shows the translation of the interrupt mode bits.

Channel type mode select - Parameter bits 2^{14} through 2^{15} are the mode bits that select the type of channel operation to be used. The selection is shown in table 7-25.

Table 7-23. Command Chaining Mode Selection

Parameter Bits 2^{11} - 2^{10}	Selection
0 0	No chaining
0 1	Chain if channel-end status is detected
1 0	Chain if device-end status is detected
1 1	Chain if either channel-end status or device-end status is detected

Table 7-24. Interrupt Mode Selection

Parameter Bits 2^{13} - 2^{12}	Selection
0 0	No interrupt mode selected, interrupts disabled
0 1	Interrupt on request-in
1 0	Interrupt on status-in
1 1	Interrupt on disconnect-in

Table 7-25. Channel Type Mode Selection

Parameter Bits 2^{15} - 2^{14}	Selection
0 0	Selector channel
0 1	Byte multiplexer channel
1 0	Block multiplexer channel
1 1	Reserved

BMA : 17 - ENTER OUTPUT TAGS

Function BMA : 17 enters the accumulator content into the output tags register, allowing direct program control of the output tags for special control sequences such as diagnostics. The Channel Busy and Channel Done flags are not altered by this function. Table 7-26 shows the accumulator bits for each output tag.

Table 7-26. Output Tags Register Bits

Accumulator Bit	Meaning
2^0	Operational out
2^1	Address out
2^2	Hold out
2^3	Select out
2^4	Command out
2^5	Service out
2^6	Data out
2^7	Suppress out
2^8	Metering out
2^9	Mark 0 out
2^{10}	Unused
2^{11}	Unused
2^{12}	Clock out
2^{13}	Inhibit parity error
2^{14}	Force BUS-OUT parity
2^{15}	Unused

PROGRAMMING EXAMPLES

The following examples illustrate two programming sequences for the block multiplexer channel. Example 1 shows the function sequence used to read a tape record of unknown length. Example 2 shows how to rewind the tape.

Example 1:

Accumulator$_8$	Function	Description
000000	BMA : 14	Set IOMA = 0 (arbitrary)
000000	BMA : 15	Set byte count = 0
000025	BMA : 16	Set device address (0-255)
000323	BMA : 2	Select GCR tape mode
	Wait for "Done"	
BUFA+2	BMA : 14	Set IOMA, chaining flag Buffer A
BUFB+3	BMA : 14	Set IOMA, chaining flag Buffer B
000005	BMA : 15	Set byte count Buffer A (header)
001000	BMA : 15	Set byte count Buffer B (1-65535)
000002	BMA : 2	Function read forward command
	Wait for interrupt	
BUFA+2	BMA : 14	Set IOMA, chaining flag, Buffer A
001000	BMA : 15	Set byte count Buffer A (1-65535)
	Wait for interrupt	Process header
BUFB+3	BMA : 14	Set IOMA, chaining flag Buffer B
001000	BMA : 15	Set byte count Buffer B
	Wait for interrupt	Transfer Buffer B to Buffer Memory
BUFA+2	BMA : 14	Set IOMA, chaining flag Buffer A
001000	BMA : 15	Set byte count Buffer A

Example 1 (continued):

Accumulator$_8$	Function	Description
		(1-65535)
	Wait for interrupt	Transfer Buffer A to Buffer Memory
	"Done" set	
	BMA : 11	
1000-X	BMA : 11	Read byte counter for residue transfer X bytes, Buffer B to Buffer Memory
	BMA : 12	
	BMA : 12	Verify device status
	BMA : 13	
	BMA : 13	Check input P.E. etc.

Example 2:

Accumulator$_8$	Function	Description
000000	BMA : 14	Set IOMA = 0 (arbitrary)
000000	BMA : 15	Set byte count = 0
000025	BMA : 16	Set device address (0-255)
000007	BMA : 2	REWIND function
	Wait for "Done"	
	Check for "Busy"	"Busy" set indicates error
	BMA : 12	
	BMA : 12	Verify device address and status
	BMA : 13	
	BMA : 13	Check for other errors

CHANNEL INTERFACE TO DISK STORAGE UNITS

Information about the channel interface to disk storage units is in Disk Systems Hardware Reference Manual, CRI publication HR-0077.

BUFFER MEMORY 8

Buffer Memory is a random access, solid-state memory designed for holding data to be transferred between peripheral devices and Central Memory or a Solid-state Storage Device (SSD). Buffer Memory capacity is 1 million, 4 million, or 8 million 64-bit words. Data is refreshed every 2 ms. Refreshing is transparent and does not affect the random access capability, although it can cause bank conflicts. Buffer Memory cycle time (the waiting time required after referencing a storage location before that location can be referred to again) is 32 clock periods (CPs).

Buffer Memory is housed in the same chassis as the I/O Processors (IOPs) to keep the intercabling delays short. This section describes Buffer Memory organization, access, addressing, and error protection.

MEMORY ORGANIZATION

Two options are available for Buffer Memory organization: Buffer Memory with 1 million or 4 million words uses an 8-bank organization; Buffer Memory with 8 million words uses a 16-bank organization. The advantages of the 16-bank organization are that the memory bandwidth is doubled and the number of bank conflicts (more than one reference to the same bank at the same time, causing one reference to wait) is greatly reduced.

Each address in Buffer Memory contains 64 data bits and 8 error correction information bits. Data is transferred to and from Buffer Memory in 16-bit parcels. Four parcels are stored in one 64-bit word. Parcels are received and sent in a 0, 1, 2, 3 sequence. The packing arrangement is shown in figure 8-1.

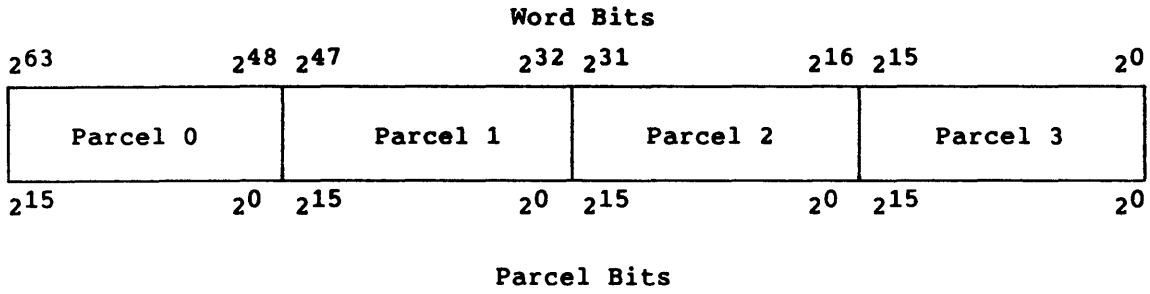


Figure 8-1. Parcel Packing in Buffer Memory Word

MEMORY ACCESS

Buffer Memory has four ports, each capable of connecting to an IOP, as shown in figure 8-2. An interface adapts one IOP Buffer Memory direct memory access (DMA) port to one Buffer Memory port. Each of the four Local Memory ports has access to all banks through a time-sharing scanner. If one port requests a reference to a bank that is busy, a reservation is made for the new port to gain access to the bank as soon as the bank becomes available. A similar scanning arrangement is used for Local Memory to share the six DMA ports.

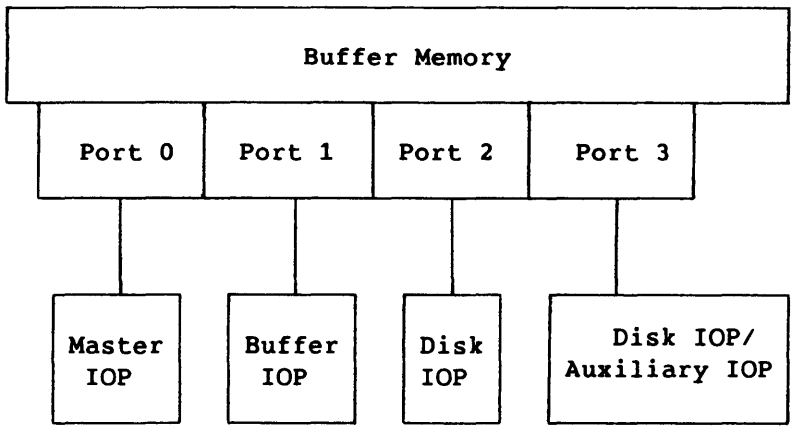


Figure 8-2. Buffer Memory Port Assignments

Buffer Memory access time is 16 CPs. However, communication speed depends on the number of banks in Buffer Memory, the activity competing for Local Memory, and the number of other IOPs attempting to use Buffer Memory.

MEMORY ADDRESSING

Buffer Memory capacity is either 1,048,576, 4,194,304, or 8,388,608 words and requires an address width of 23 bits. The 23-bit address is provided from an IOP accumulator in two functions to the interface as shown in figure 8-3. Buffer Memory does not address to the parcel level. The address represents a physical location in Buffer Memory.

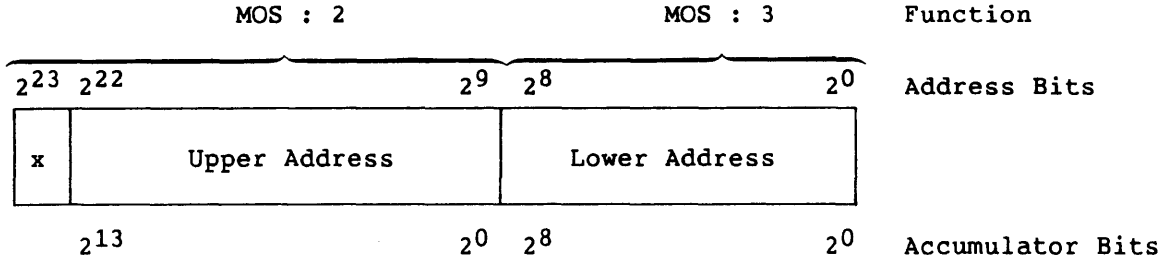


Figure 8-3. Buffer Memory Address Formation

ERROR PROTECTION

Buffer Memory uses single-error correction/double-error detection (SECDED) logic to determine if the data has been altered by the storage cycle. When the data is written into Buffer Memory, a checkword is generated for the word and stored with that word. The checkword is an 8-bit Hamming† code. When the word is read from Buffer Memory, the checkword and data word are processed to determine if any bits were altered. If no errors occurred, the word is passed to the IOP.

If an error did occur, the 8 bits of the checkword are analyzed by the logic to find out if only 1 bit has been altered or if more than 1 bit has changed. If only 1 bit has been altered, the correction logic resets that bit to the correct state and passes the corrected word out to the IOP.

If more than 1 bit of the stored word has been altered, the logic cannot correct the word. The interface signals the error condition by leaving the Channel Busy signal set after the block transfer, interrupting the IOP for an error handling routine. The Master I/O Processor (MIOP) receives an interrupt on the error logging channel. The error handling routine can include a call to the MIOP to have that processor retrieve the error information. Error handling is explained in detail in section 7.

† Hamming, R. W. "Error Detection and Correcting Codes." *Bell System Technical Journal* 29, No. 2 (April 1950) 147-160.

Buffer Memory uses the same SECDED logic design as used in the Cray mainframe Central Memory. Refer to the mainframe reference manuals listed in the preface.

APPENDIX SECTION

I/O PROCESSOR INSTRUCTION SUMMARY

IOP	APML	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 *d* to A
021	A = A & *dd*	Logical product of A and operand register *d* to A
022	A = A + *dd*	Add operand register *d* to A
023	A = A - *dd*	Subtract operand register *d* from A
024	*dd* = A	Transmit A to register *d*
025	*dd* = A + *dd*	Add operand register *d* to A, result to operand register *d*
026	*dd* = *dd* + 1	Transmit register *d* to A, add 1, result to operand register *d*
027	*dd* = *dd* - 1	Transmit register *d* to A, subtract 1, result to operand register *d*
030	A = (*dd*)	Transmit contents of memory addressed by register *d* to A
031	A = A & (*dd*)	Logical product of A and contents of memory addressed by register *d*, result to A
032	A = A + (*dd*)	Add contents of memory addressed by register *d* to A, result to A
033	A = A - (*dd*)	Subtract contents of memory addressed by register *d* from A, result to A

IOP	APML	Description
034	(*dd*) = A	Transmit A to memory addressed by register *d*
035	(*dd*) = A + (*dd*)	Add memory addressed by register *d* to A, result to same memory location
036	(*dd*) = (*dd*) + 1	Transmit memory addressed by register *d* to A, add 1, result to same memory location
037	(*dd*) = (*dd*) - 1	Transmit memory addressed by register *d* to A, subtract 1, result to same memory location
040	C = 1, *iod* = DN	Set carry equal to channel *d* done
041	C = 1, *iod* = BZ	Set carry equal to channel *d* busy
042	C = 1, IOB = DN	Set carry equal to channel B done
043	C = 1, IOB = BZ	Set carry equal to 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, result to A
053	A = A - B	Subtract B from A, result to 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, result to A
063	A = A - (B)	Subtract operand register B from A, result to A
064	(B) = A	Transmit A to operand register B
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	P = P + *d*	Return jump to P + *d*
073	P = P - *d*	Return jump to P - *d*
074	P = *dd*	Jump to address in operand register *d*
075	P = *dd* + *k*	Jump to sum of *k* and operand register *d*
076	R = *dd*	Return jump to address in operand register *d*
077	R = *dd* + *k*	Return jump to sum of *k* and operand register *d*

IOP	APML	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 *d* if carry = 0
121	P = *dd*, C # 0	Jump to address in operand register *d* if carry ≠ 0
122	P = *dd*, A = 0	Jump to address in operand register *d* if A = 0
123	P = *dd*, A # 0	Jump to address in operand register *d* if A ≠ 0
124	P = *dd* + *k*, C = 0	Jump to address in operand register *d* + *k* if carry = 0
125	P = *dd* + *k*, C # 0	Jump to address in operand register *d* + *k* if carry ≠ 0
126	P = *dd* + *k*, A = 0	Jump to address in operand register *d* + *k* if A = 0
127	P = *dd* + *k*, A # 0	Jump to address in operand register *d* + *k* if A ≠ 0
130	R = *dd*, C = 0	Return jump to address in operand register *d* if carry = 0
131	R = *dd*, C # 0	Return jump to address in operand register *d* if carry ≠ 0
132	R = *dd*, A = 0	Return jump to address in operand register *d* if A = 0
133	R = *dd*, A # 0	Return jump to address in operand register *d* if A ≠ 0

IOP	APML	Description
134	R = *dd* + *k*, C = 0	Return jump to address in operand register *d* + *k* if carry = 0
135	R = *dd* + *k*, C # 0	Return jump to address in operand register *d* + *k* if carry ≠ 0
136	R = *dd* + *k*, A = 0	Return jump to address in operand register *d* + *k* if A = 0
137	R = *dd* + *k*, A # 0	Return jump to address in operand register *d* + *k* if A ≠ 0
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
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
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
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
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
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

I/O PROCESSOR PROGRAMMING CONSIDERATIONS

B

This section summarizes special cases that must be considered when programming an I/O Processor (IOP).

EXIT STACK TIMING

When issuing PXS : 14 or PXS : 15, allow 5 clock periods (CPs)† for enabling system interrupts or issuing return jumps or exit instructions. Instructions PXS : 14 and PXS : 15 should only be used when system interrupts are disabled.

EXIT STACK INTERRUPT HANDLING

If return jumps are used in an interrupt handler, verify that enough levels are left available in the stack. An interrupt with the exit stack pointer at 13 causes the pointer to go to 14 and leaves only one location open. A worse case exists if a return jump which causes a program fetch request (PFR) interrupt is issued with the stack pointer at 13. The return address goes into 14 and the interrupt address goes into 15. This condition now leaves two interrupts present: the exit stack boundary and PFR (with the PFR being the highest priority). The exit stack now wraps around to 0 and the address that was at exit stack location 0 is corrupted.

SYSTEM INTERRUPT ENABLE

When issuing instruction 003 (I = 1), the system interrupt enable is delayed until the next nonbranch or non-I/O instruction is issued. Instructions 40 through 43 and 70 through 137 do not enable interrupts.

† The term clock period refers to processor instruction time for issuing pass instructions or other instructions or group of instructions whose execution time equals or exceeds the delays noted. Any instruction or group of instructions can be used as long as it is not included in any of the special cases.

This allows the executive/monitor to get back to the interruptible activity before an interrupt is accepted.

Instruction I = 0 should be used at the interrupt handler entrance. If a redundant instruction I = 1 is executed and an interrupt occurs before a nonbranch or non-I/O instruction is encountered, the interrupt handler is entered (with interrupts disabled). However, interrupts are reenabled when the first nonbranch or non-I/O instruction is issued within the interrupt handler.

SYSTEM INTERRUPT DISABLE

The instructions following an instruction 002 (I = 0) can be skipped if an interrupt occurs (while instruction 002 is executing). Therefore, a pass instruction should follow every instruction 002.

SYSTEM INTERRUPT CLEARED OR SET BY ENABLES FOR INDIVIDUAL CHANNELS

After issuing a channel command 6 or 7 to any I/O channel, allow 3 CPs before seeing its effect on system interrupt. (Assuming system interrupts are, or will be, enabled.)

I/O CHANNEL TIMING

When issuing any command to the I/O channels, allow 1 CP before checking Busy or Done status.

Also allow 1 CP after any channel command 6 or 7 before checking for interrupt number (IOR : 10).

BUFFER MEMORY ERRORS

If a Buffer Memory multiple-bit error has occurred, instruction MOS : 0 is required prior to the next read or write command. The interface operation waits indefinitely on the error if it is not cleared by instruction MOS : 0.

BUFFER MEMORY DEADSTART TIME

Deadstarting a processor through Buffer Memory requires approximately 2 ms to transfer the full 64K into Local Memory. This time assumes no other Buffer Memory activity other than refresh.

ERROR LOGGING AND BLOCK MULTIPLEXER CHANNELS (SERIAL NO. 20 AND BELOW)

The commands ERA : 10 through 13 to the error logging channel or the block multiplexer channel decode the present accumulator data on the interface. If the IOP instruction previous to the commands ERA : 10 through 13 is 4 through 7, 12, 13, 16, 17, 22, 23, 32, 33, 44 through 47, 52, 53, 62, or 63, then instruction 11 or 15 (with the *d* or *k* field set to all ones) should be inserted between the instruction and the interface command. This avoids a 1-CP timing restriction caused by the Adder and Shifter.

I/O INSTRUCTIONS AFTER DEADSTART

The first instruction executed after a deadstart cannot be an I/O instruction (such as, instructions 40 through 43, 140 through 147, 154 through 157, 160 through 167, and 174 through 177). The accumulator must be loaded before executing any of these instructions to avoid a special control sequence condition after deadstart.

NOTE

Instructions 150 through 153 and 170 through 173 load the accumulator and can be executed after deadstart.

PERIPHERAL EXPANDER CHANNEL TRANSFERS

While the Peripheral Expander Channel supports block transfers for the full $200{,}000_8$ parcels of Local Memory, the tape and printer controllers are limited to reading and writing $100{,}000_8$ parcels.

SYSTEM CHANNEL ASSIGNMENTS C

The channel assignments for a typical CRAY-1, model S/4400 Computer System are shown in table C-1.

Table C-1. Typical Model S/4400 System Channel Assignments

Processor	Channel	Mnemonic	Function
Master I/O Processor	0	IOR	Interrupt request
	1	PFR	Program fetch request
	2	PXS	Program exit stack
	3	LME	Local Memory error
	4	RTC	Real-time clock
	5	MOS	Buffer Memory interface (DMA 3)
	6	AIA	Input from Buffer I/O Processor
	7	AOA	Output to Buffer I/O Processor
	10	AIB	Input from Disk I/O Processor
	11	AOB	Output to Disk I/O Processor
	12	AIC	Input from Auxiliary/Disk I/O Processor
	13	AOC	Output to Auxiliary/Disk I/O Processor
	14		
	15		
	16	ERA	Error log
	17	EXB	Peripheral Expander (DMA 0)
	20	CIA	Input from Cray Computer System channel (DMA 1)
	21	COA	Output to Cray Computer System channel (DMA 1)
	22		
	23		
	24	CIB	Input from front-end interface (DMA 2)
	25	COC	Output to front-end interface (DMA 2)
	26		
	27		
	30	CIC	Input from front-end interface (DMA 4)
	31	COC	Output to front-end interface (DMA 4)
	32		
	33		
	34	CID	Input from front-end interface (DMA 5)
	35	COD	Output to front-end interface (DMA 5)

Table C-1. Typical Model S/4400 System Channel Assignments (continued)

Processor	Channel	Mnemonic	Function
	36		
	37		
Master	40	TIA	Console 0 keyboard
I/O	41	TOA	Console 0 display
Processor	42	TIB	Console 1 keyboard
(continued)	43	TOB	Console 1 display
	44		
	45, 46, 47		Available for other consoles
Buffer	0	IOR	Interrupt request
I/O	1	PFR	Program fetch request
Processor	2	PXS	Program exit stack
	3	LME	Local Memory error
	4	RTC	Real-time clock
	5	MOS	Buffer Memory interface (DMA 3)
	6	AIA	Input from Master I/O Processor
	7	AOA	Output from Master I/O Processor
	10	AIB	Input from Disk I/O Processor
	11	AOB	Output to Disk I/O Processor
	12	AIC	Input from Auxiliary/Disk I/O Processor
	13	AOC	Output to Auxiliary/Disk I/O Processor
	14	HIA	Input from Memory (100 Mbyte) Channel or SSD (DMA 4)
	15	HOA	Output to Memory (100 Mbyte) Channel or SSD (DMA 4)
	16		
	17		
	20	DKA†	Disk storage unit 0 (DMA 0)
	21	DKB	Disk storage unit 1 (DMA 0)
	22	DKC	Disk storage unit 2 (DMA 0)
	23	DKD	Disk storage unit 3 (DMA 0)
	24	DKE	Disk storage unit 4 (DMA 1)
	25	DKF	Disk storage unit 5 (DMA 1)
	26	DKG	Disk storage unit 6 (DMA 1)
	27	DKH	Disk storage unit 7 (DMA 1)
	30	DKI	Disk storage unit 8 (DMA 2)
	31	DKJ	Disk storage unit 9 (DMA 2)
	32	DKK	Disk storage unit 10 (DMA 2)
	33	DKL	Disk storage unit 11 (DMA 2)
	34	DKM	Disk storage unit 12 (DMA 5)

† If the DSU interfaces to the DCU-5 controller, the mnemonics are DIA through DIP.

Table C-1. Typical Model S/4400 System Channel Assignments (continued)

Processor	Channel	Mnemonic	Function
Buffer I/O Processor (continued)	35	DKN	Disk storage unit 13 (DMA 5)
	36	DKO	Disk storage unit 14 (DMA 5)
	37	DKP	Disk storage unit 15 (DMA 5)
	40	TIA	Console 0 keyboard
	41	TOA	Console 0 display
	42		Available for other consoles
	43		
	44		
	45		
	46		
	47		
Disk I/O Processor	0	IOR	Interrupt request
	1	PFR	Program fetch request
	2	PXR	Program exit stack
	3	LME	Local Memory error
	4	RTC	Real-time clock
	5	MOS	Buffer Memory interface (DMA 3)
	6	AIA	Input from Master I/O Processor
	7	AOA	Output to Master I/O Processor
	10	AIB	Input from Buffer I/O Processor
	11	AOB	Output to Buffer I/O Processor
	12	AIC	Input from Auxiliary/Disk I/O Processor
	13	AOC	Output to Auxiliary/Disk I/O Processor
	14	HIA	Input from Memory (100 Mbyte) Channel or SSD (DMA 4)
	15	HOA	Output to Memory (100 Mbyte) Channel or SSD (DMA 4)
	16		
	17		
	20	DKA†	Disk storage unit 0 (DMA 0)
	21	DKB	Disk storage unit 1 (DMA 0)
	22	DKC	Disk storage unit 2 (DMA 0)
	23	DKD	Disk storage unit 3 (DMA 0)
	24	DKE	Disk storage unit 4 (DMA 1)
	25	DKF	Disk storage unit 5 (DMA 1)
	26	DKG	Disk storage unit 6 (DMA 1)
	27	DKH	Disk storage unit 7 (DMA 1)
	30	DKI	Disk storage unit 8 (DMA 2)
	31	DKJ	Disk storage unit 9 (DMA 2)
	32	DKK	Disk storage unit 10 (DMA 2)
	33	DKL	Disk storage unit 11 (DMA 2)

† If the DSU interfaces to the DCU-5 controller, the mnemonics are DIA through DIP.

Table C-1. Typical Model S/4400 System Channel Assignments (continued)

Processor	Channel	Mnemonic	Function
Disk I/O Processor (continued)	34	DKM	Disk storage unit 12 (DMA 5)
	35	DKN	Disk storage unit 13 (DMA 5)
	36	DKO	Disk storage unit 14 (DMA 5)
	37	DKP	Disk storage unit 15 (DMA 5)
	40	TIA	Console 0 keyboard
	41	TOA	Console 0 display
	42		Available for other consoles
	43		
	44		
	45		
	46		
	47		
Auxiliary I/O Processor	0	IOR	Interrupt request
	1	PFR	Program fetch request
	2	PXS	Program exit stack
	3	LME	Local Memory error
	4	RTC	Real-time clock
	5	MOS	Buffer Memory interface (DMA 3)
	6	AIA	Input from Master I/O Processor
	7	AOA	Output to Master I/O Processor
	10	AIB	Input from Buffer I/O Processor
	11	AOB	Output to Buffer I/O Processor
	12	AIC	Input from Disk I/O Processor
	13	AOC	Output to Disk I/O Processor
	14		
	15		
	16		
	17		
	20	BMA	Block multiplexer channel 0 (DMA 0)
	21	BMB	Block multiplexer channel 1 (DMA 0)
	22	BMC	Block multiplexer channel 2 (DMA 0)
	23	BMD	Block multiplexer channel 3 (DMA 0)
	24	BME	Block multiplexer channel 4 (DMA 1)
	25	BMF	Block multiplexer channel 5 (DMA 1)
	26	BMG	Block multiplexer channel 6 (DMA 1)
	27	BMH	Block multiplexer channel 7 (DMA 1)
	30	BMI	Block multiplexer channel 8 (DMA 2)
	31	BMJ	Block multiplexer channel 9 (DMA 2)
	32	BMK	Block multiplexer channel 10 (DMA 2)
	33	BML	Block multiplexer channel 11 (DMA 2)
	34	BMM	Block multiplexer channel 12 (DMA 5)
	35	BMN	Block multiplexer channel 13 (DMA 5)
	36	BMO	Block multiplexer channel 14 (DMA 5)
	37	BMP	Block multiplexer channel 15 (DMA 5)

Table C-1. Typical Model S/4400 System Channel Assignments (continued)

Processor	Channel	Mnemonic	Function
Auxiliary I/O Processor (continued)	40	TIA	Console 0 keyboard
	41	TOA	Console 0 display
	42		Available for other consoles
	43		
	44		
	45		
	46		
	47		

ABBREVIATIONS D

A, An	IOP accumulator
Addr	Address
Adv.	Advance
A*i*,A*j*,A*k*	Address register specified by instruction *i*, *j*, and *k* fields
APML	A Programming Machine Language
B,B*n*	IOP B register
BA	Bank address, buffer address
BIOP	Buffer I/O Processor
B*jk*	Buffer register specified by instruction *j* and *k* fields
BM	Buffer Memory
CA	Current address register
CAL	Cray Assembly Language
Ch	Channel
CIP	Current Instruction Parcel register
CL	Channel Limit register
CLK	Clock
Contr.	Control
CP	Clock period, central processor
CPU	Central processing unit
CRI	Cray Research, Incorporated
d	*d* field
DCU	Disk controller unit
DIOP	Disk I/O Processor
Distr.	Distribution
DMA	Direct memory access
DP	Destination Pointer register
DSU	Disk storage unit
Exch.	Exchange
f	*f* field

F	Flag register (Exchange Package)
F.E.	Front end, field engineer
F.P.	Floating-point
F.U.	Functional Unit
FWA	First word address
gh	*g* and *h* fields, CPU instruction operation code
GR	Group
h	*h* field, CPU instruction
Hz	Hertz, cycles per second
i	*i* field of CPU instruction
IC	Integrated Circuit
ICD	Interrupt Countdown counter
II	Interrupt Interval register (CPU); Instruction Issue register (IOP)
I/O	Input/Output
IOP	I/O Processor
IOR	I/O request
IOS	I/O Subsystem
k	Kilo, 1024, *k* field CPU instruction
jk	*j* and *k* fields, CPU instruction
LA	Limit address
LIP	Last instruction parcel
LSI	Large scale integration
LWA	Last word address
M	Million; mode bit field in Exchange Package; instruction field.
Mbits	Megabits or million bits
Mbyte	Megabyte or million bytes
MCU	Maintenance Control Unit
MG	Motor-generator
MHz	MegaHertz, or million cycles per second
MIOP	Master I/O Processor
MMI	Monitor Mode Interrupt
ms	Millisecond
MSKO	Mask out

NIP	Next instruction parcel
ns	Nanosecond
OS	Operating system
P	Program address register; Program parcel counter
PCI	Programmable clock interrupt
PDU	Power Distribution Unit
PFR	Program fetch request
POP	Population count
R	Request; response
RA	Read address
Recip	Reciprocal
Ref.	Reference
Reg.	Register
Req.	Request
Resp.	Response
RP	Register Pointer register
R'RAB	R' = high-order bits of read address, RA = Read Address, B = Bank low-order bits of address in exchange package
RTC	Real-time clock
s	Second
S	Scalar
SECDED	Single error correction/double error detection
Seq.	Sequence
S*n*	Scalar register, *n* = 0 to 7
S*i*,S*j*,S*k*	Scalar register specified by instruction *i* or *j* or *k* field
Stor.	Storage
SSD	Solid-state Storage Device
T, T*n*	Intermediate scalar register, *n* = 0 to 77_8
T*jk*	Temporary register indicated by instruction *j*,*k* fields
V, V*n*	Vector register, *n* = 0 to 7
V*i*,V*j*,V*k*	Vector register specified by instruction *i* or *j* or *k* field
VL	Vector Length register
VM	Vector Mask register
XA	Exchange Address register
XIOP	Auxiliary I/O Processor

INDEX

INDEX

READER COMMENT FORM

I/O Subsystem Model B Hardware Reference Manual HR-0030 B

Your comments help us to improve the quality and usefulness of our publications. Please use the space provided below to share with us your comments. When possible, please give specific page and paragraph references.

NAME _____

JOB TITLE _____

FIRM _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

DATE _____

CRAY RESEARCH, INC.

CUT ALONG THIS LINE

FOLD

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.

Attention: PUBLICATIONS
Technical Operations Building
890 Industrial Boulevard
Chippewa Falls, WI 54729

FOLD

STAPLE

READER COMMENT FORM

I/O Subsystem Model B Hardware Reference Manual HR-0030 B

Your comments help us to improve the quality and usefulness of our publications. Please use the space provided below to share with us your comments. When possible, please give specific page and paragraph references.

NAME _____

JOB TITLE _____

FIRM _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

DATE _____

CRAY RESEARCH, INC.

CUT ALONG THIS LINE

FOLD

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.

Attention: PUBLICATIONS
Technical Operations Building
890 Industrial Boulevard
Chippewa Falls, WI 54729

FOLD

STAPLE